Your Testing is a Joke!

EuroSTAR 'Best Paper' Winner 2015

James Thomas

Linguamatics, UK

TESTHuddle

EuroSTAR
Software Testing
CONFERENCE

Is This Book Right For Me?

INTRODUCTORY

Introductory content is for software testing professionals who are relatively new to the subject matter of the ebook. Introductory ebooks typically feature an overview of an aspect of testing or guidance on understanding the fundamentals of the subject at hand.

INTERMEDIATE

Intermediate ebooks are for software testers who are already familiar with the subject matter of the eBook but have limited hands-on experience in this area. Intermediate level usually focuses on working examples of a concept or relaying experiences of applying the approach in a particular context.

ADVANCED

Advanced ebook content is for software testers who are, or intend to become, experts on the subject matter. These ebooks look at more advanced features of an aspect of software testing and help the reader to develop a more complete understanding of the subject.

Abstract

Edward de Bono, in his Lateral Thinking books, makes a strong connection between humour and creativity. Creativity is key to testing, but jokes? Well, the punchline for a joke could be a violation of some expectation, the exposure of some ambiguity, an observation that no one else has made, or just making a surprising connection. Jokes can make you think and then laugh. But they don't always work. Does that sound familiar?

This eBook takes a genuine joke-making process and deconstructs it to make comparisons between aspects of joking and concepts from testing such as the difference between a fault and a failure, oracles, heuristics, factoring, modelling testing as the exploration of a space of possibilities, stopping strategies, bug advocacy and the possibility that a bug, today, in this context might not be one tomorrow or in another.

It goes on to wonder about the generality of the observations and what the value of them might be before suggesting ways in which joking can provide useful practice for testing skills.

There are some jokes in the eBook, of course. And also an explanation of why any groaning they provoke is a good sign…

Biography

James Thomas Linguamatics, UK

James is one of the founders of Linguamatics, the world leader in innovative natural language-based text mining. Over the years he's had many roles in the company and is currently the test manager, a position in which he strives to provide an environment where his testers have an opportunity to do their best work. He’s on Twitter as @qahiccupps and blogs at Hiccupps.

Contents

Your Testing is a Joke

Your testing is a joke. Yeah, you read that right; I'm saying your testing is a joke.

But what I mean is that some parts of some of the testing done by some people reading this will be somewhat analogous to some subset of what some people would accept as jokes. Sometimes.

Language can be tricky like that. And that's one of the things that makes it such a productive tool for jokes and a tricky tool for software development. Even the word joke *itself* is ambiguous, having senses related to amusement, pathetic inadequacy and seriousness. Used together, these make it possible for me to say, simultaneously self-referentially, tautologically and contradictorily:

> *My joke is a joke and it's no joke.*

I find interesting parallels between the activities of testing and joke-making. In this paper I'll try to explain how and why and what some of the side-effects or benefits of thinking this way are, with particular reference to these three concepts:

- Surprise: the identification of some aspect of a situation which is incongruous.
- Coherence: the presentation of those observations in a relevant context.
- Delivery: making a convincing story about surprise and coherence and telling it well.

What is a Joke?

According to Oxford Dictionaries, the primary definition of joke[1] is a thing that someone says

> *to cause amusement or laughter, especially a story with a funny punch line*

And humans like amusement and laughter. Edward de Bono is quoted as saying[2] "Humour is by far the most significant activity of the human brain." In **LATERAL THINKING** he makes strong connections between various kinds of creativity, including humour:

> *Lateral thinking is closely related to insight, creativity and humour [...] Insight, creativity and humour are so elusive because the mind is so efficient. The mind functions to create patterns out of its surroundings ... As the patterns are used they become ever more firmly established.*

Hurley, Dennett and Adams, in **INSIDE JOKES**, propose that humour evolved from a basic human need for consistency-checking of mental models generated by the same kind of unconscious cognitive pattern-matching that de Bono refers to:

> *Undersupervised and of variable reliability [the models'] contributions need to be subjected to frequent "reality checks"*

The argument runs something like this: we are conditioned to let our brain just "take care" of much of our day-to-day existence, making decisions based on what *usually happens* in a given situation. Mostly this will work out fine but occasionally it does not. Imagine our early ancestors hearing a twig snap in a jungle while hunting. Usually this will be nothing important but now and again it will be a

[1] http://www.oxforddictionaries.com/definition/english/joke
[2] http://en.wikiquote.org/wiki/Edward_de_Bono

big and scary predator and they'll need to realise that their unconscious assumption, the expectation that "it was nothing", is wrong and they should run away. Very quickly.

Likewise, the punch line for a joke is frequently also a violation of some expectation, either about the world or the context set up by the joke itself. Hurley et al's theory runs along the lines that the same consistency checking mechanism is what finds that discrepancy in the listener's mental model and provokes the humour.

And why might this interest testers?

Well, this joke from Stewart Lee's **HOW I ESCAPED MY CERTAIN FATE** illustrates the pull back and reveal[3] pattern where an expectation is set up and then torn down:

> *Every day at school I used to get bullied, kicked, spat at and pushed into a urinal … so after a while I resigned from my job as headmaster.*

Software is built for people and people have expectations, and contexts, which are populated both consciously and unconsciously. Adding information to those contexts can expose them as invalid. Testers can do that.

In some cases, comedy comes from recognition, from the familiar being recast in a new light. This is known as observational comedy and George Carlin is a master of it. This is one of his[4]:

> *If the "black box" flight recorder is never damaged during a plane crash, why isn't the whole damn airplane made out of that shit?*

Often testers will want to observe things that others have passed over or just become inured to – the usability glitch that makes a conceptually simple task into a complex sequence of actions, for example. Once outed, it can become an issue for discussion.

Ambiguity can be a nightmare in software development – the two stakeholders that use the same words to mean two very different things or, perhaps worse, two subtly different things. And it powers jokes too; try this old one:

> *Jack: My dog's got no nose*
> *Jill: Really, how does it smell?*
> *Jack: Absolutely terrible.*

Another kind of joke, the pun, gets its punch line by making an unexpected connection or juxtaposition. I love puns and, for me, the *cornier* the better:

- When I have a really corny pun to deliver, I use my huskiest voice.
- Some people hate corny puns. I tell them not to get a cob on.
- But some people I know also love corny puns, and that's sweet.

Figure 1 shows the kinds of connections being made here. Sometimes, of course, the connection or ambiguity or assumption is opaque to the listener, perhaps because the context is wrong or they

[3] http://EzineArticles.com/3177289
[4] http://www.jokes4us.com/peoplejokes/comedianjokes/georgecarlinjokes.html

have insufficient domain knowledge to perceive it – the expression “get a cob on” is quite colloquial English, for instance – and in this case, they might not “get” the joke.

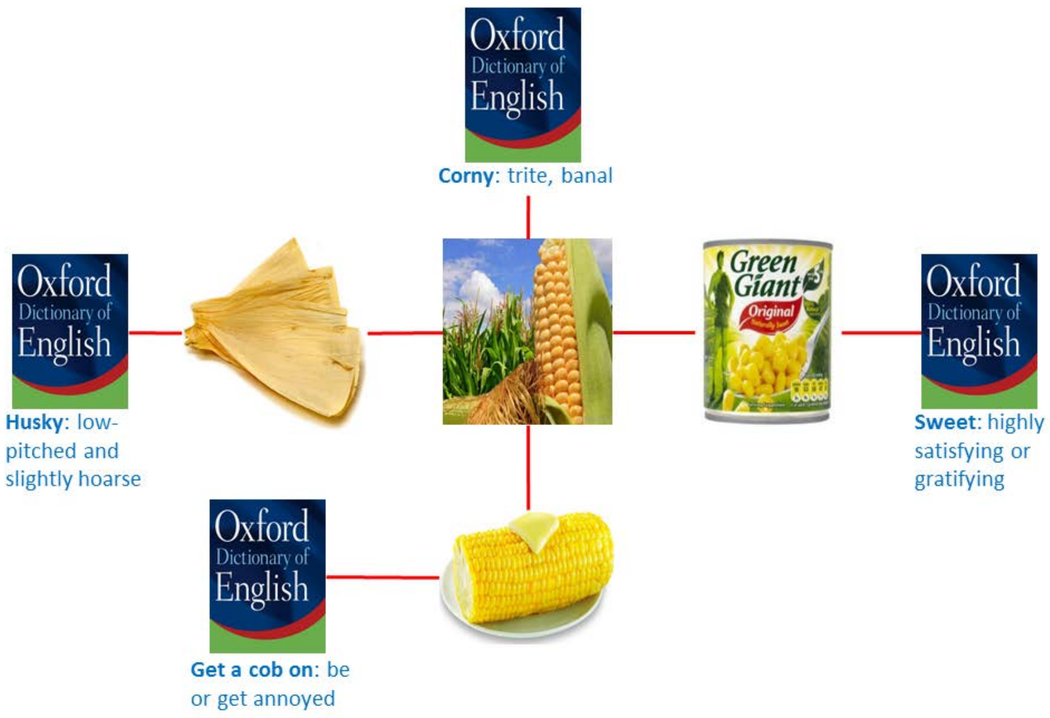
Oxford Dictionary of English
Corny: trite, banal
Oxford Dictionary of English
Husky: low-pitched and slightly hoarse
Green Giant Original
Oxford Dictionary of English
Sweet: highly satisfying or gratifying
Oxford Dictionary of English
Get a cob on: be or get annoyed

Figure 1: Corny connections for punning

Brownell and Gardner, in their essay in **LAUGHING MATTERS**, break jokes down as having two formal components:

- surprise: say, by a violated expectation.
- coherence: how well the joke fits to the original context.

This seems quite plausible given what we’ve just discussed. Note that surprise without coherence[5] gives us a non sequitur[6]. This violates expectation but has no fit to the context:

> *Q: Why was the programmer debugging at midnight?*
> *A: Fish.*

[5] There are four permutations of the two variables, surprise and coherence. Two are discussed in the main text. For completeness, here are the other two.

Coherence without surprise is not generally a joke: “I walked into the bank today, said hello to the cashier, and withdrew some money. What a pleasant and efficient transaction that was.”

A lack of both surprise and coherence generally doesn’t provoke laughter either, as in this famous sentence from Noam Chomsky: “Colourless green ideas sleep furiously.”

[6] A statement that doesn’t follow from the preceding statement, e.g. http://www.oxforddictionaries.com/definition/english/non-sequitur

Of course, there are surrealist comedians who trade in this kind of humour[7] which suggests that the surprise/coherence concept is heuristic: useful but also fallible.

So, again, why would a tester be interested in joking? Perhaps because it involves violated expectations, ambiguities, exposed assumptions, checking, making connections and is heavily dependent on context?

I find that, intuitively and with introspection, the kinds of things I do when creating a joke overlap with the kinds of things I do while testing. I'm interested to know what might be learned from pursuing an analogy between testing and joking; what techniques might cross over, what heuristics there are in comedy that could be applied to testing – and vice versa – and also in having a laugh.

Joking and Testing

In January 2015 I saw this on Twitter[8]:

> *Q: Why was the web developer sent home early on new years?*
> *A: His application needed a little REST.*

By Brownell and Gardner's analysis, this constitutes a joke. The setup (Q) defines the concepts in the domain while the punch line (A) gets its surprise from the ambiguity between REST (Representational State Transfer, a web architecture concept) and rest (being tired). The coherence comes from both of those concepts being relevant in the context of Q.

To help explore the ideas I was having about the relationship between joking and testing at the time, I gave myself the missions of (1) testing the joke and (2) generating new jokes from the same set up, and recording what I was thinking and why. This section describes some of what I did, in the order I did it.

Testing the Joke

To start with, I applied a *sufficiency/necessity* filter; something I often do when testing. The punch line doesn't refer to the New Year's Eve part of the setup and, if we were interested in refining the joke, we might recast it as:

> *Q: Why was the web developer sent home early?*
> *A: His application needed a little REST.*

Consistency is a watchword for testers and there's a potential inconsistency here: the *developer* was sent home, but *the application* needed REST. Since REST is a technical concept the joke self-restricts its audience (or, at least, the audience who will understand it) and so we are free to craft the joke further to remove information that is *known* in that context, e.g. if he's a developer then it's safe to assume that there is an application, so we can remove the slight inconsistency:

> *Q: Why was the web developer sent home early?*
> *A: He needed REST.*

[7] Although just saying "fish" is only *cod surrealism.*
[8] https://twitter.com/Veretax/status/551013695875448833

A third tool I'll almost always apply in a testing task is *balance*: between depth and breadth of coverage; between this area and that area; between priority and severity; between usability and performance; between this project and that project; between the investigation of a candidate fix and the reporting of a new issue. And, frequently, also between many of these at once.

In this case, I wondered whether there's a better balance to be had between coherence and surprise. Do developers (or their applications) *need* REST? Is there a more ambiguous term that could retain *surprise* but increase *coherence*?

> *Q: Why was the web developer sent home early?*
> *A: He wanted REST.*

Now we have at least a couple of competing interpretations, still exploiting the original ambiguity:

- He was sent home because he was tired.
- He was sent home because he was biased towards a technology.

This is (to me) a better gag and expresses the core thought of the joke more efficiently. But, interestingly, this exposes a way in which jokes align with testing - they are subject to the Relative Rule[9]:

> *A joke is a joke to some person at some time.*
>
> *An issue is an issue to some person at some time.*

I think the gag is funnier; *I* think this report is more compelling. But your mileage may vary. And tomorrow we might each feel differently.

When I make a corny pun and get a groan in response, I know that I'm just within the bounds of plausibility for the person I'm talking to. But sometimes, your audience just *doesn't get it*. For them, maybe there is no ambiguity or they don't see why an expectation is violated.

Likewise, have you ever had to spell out just *why* something is an issue to a stakeholder? The joke dissected is rarely as funny, and it's often similar with bugs – if the person you're trying to persuade didn't see it as an issue instinctively, they might still deprioritise it after having the explanation – although when the context changes and a *customer* reports seeing the same thing, their appreciation of the issue can change rapidly.

I find that I can insulate myself against this to some extent by thinking explicitly about coherence and surprise when reporting a potential issue[10], for example:

- coherence: fit to a stakeholder scenario, use case, requirement etc.

[9] Abstract concepts are defined by their relationship to the person considering them, in the context they are being considered, e.g. http://www.developsense.com/blog/2010/09/done-the-relative-rule-and-the-unsettling-rule/

[10] See, for example, the "postage stamp bug" in Cem Kaner's Scenario Testing paper, http://www.kaner.com/pdfs/ScenarioIntroVer4.pdf, where finding the particular hot button for a particular marketing manager resulted in a bug that was previously considered unimportant being fixed with alacrity.

- surprise: extent to which the outcome is expected by the stakeholder.

Designing a Joke

In the first mission, above, I was able to use techniques that I use in testing to refine the joke. My second mission took a different tack: using the setup as a starting point for the generation of new punch lines.

> *Q: Why was the web developer sent home early on new years?*

My intuition is that this is the exploration of a search space, very parallel to how I see the act of testing. It’s a more open-ended task, somewhat like starting with a requirement and thinking about potential problems and design decisions.

This exposes a way in which jokes both are, and are not, like testing. When you know someone's telling you a joke, you know that you're (almost always) going to get a punch line (of some kind). When you start testing, however, there are no guarantees that your work will result in any issues being found.

But when you're expecting a punch line, you will probably be more (even unconsciously) creative in making connections or in accommodating them when they are delivered. So, when testing, I wonder whether it would be advantageous to put yourself in that position and *expect there is an issue for you to find*?[11]

On the other hand, from the perspective of the joke writer there are no assurances either. A starting point might be productive or unproductive, today or tomorrow, for one person or another, and so on. But "pure" joke writing doesn't necessarily start from a concrete setup. The joke writer will probably have something less concrete initially and the task for them will be to construct a setup *and* a punch line that work in harmony.

Given a set of requirements, the tester's role is frequently more reactive. But taking a guide from the comedian might naturally result in more questions about the appropriateness of the requirement, or alternatives to it.

So let's see how I reacted to the original setup:

> *Q: Why was the web developer sent home early on new years?*

Without a reason to start my testing in any particular place, I will often pick out the entities ...

- web
- developer
- web developer

[11] In the version of this paper submitted to EuroSTAR 2015 I asked for readers’ thoughts on this point. One correspondent suggested that, unlike a defendant being considered innocent until proven guilty in a court of law, our starting point in the court of software development should be that an application is *guilty*.

Another pointed me at the BBST Bug Advocacy course notes which, via Glen Myers, say “testers who want to find bugs are more likely to notice program misbehavior than people who want to verify that the program works correctly” (http://www.testingeducation.org/BBST/bugadvocacy/BugAdvocacy2008.pdf#page=130).

- new year
- home
- work (implicit)

... and then look for concepts (of any kind) related to them (in any way) ...

- Web - HTTP, SOAP, browser, web, internet, JavaScript, AJAX, …
- developer - code, script, IDE, bugs, …
- new year - 31st, 1st, annual, fireworks, resolution, …
- home - home page, house, street, homing pigeon, …
- work - job, position, meetings, requirements, responsibilities, …

... and then see whether any of those concepts could link to one another or something related. I call this *expansion* and it feels like a close cousin of factoring[12]. But where factoring is a process which seeks to list the intrinsic dimensions of a thing that may be relevant to the task in hand, expansion additionally looks for more general properties, instances, sub-parts, synonyms, antonyms and other related terms.

In my mental model, expansion takes a set of points in a space and blows them up into areas, reducing the “conceptual distance” between two things, making more opportunity for overlaps and intersections, making it less of a leap to make connections between the concepts, helping to find places where we can identify or postulate relationships, ambiguities, contradictions, and so on, as in Figure 2.

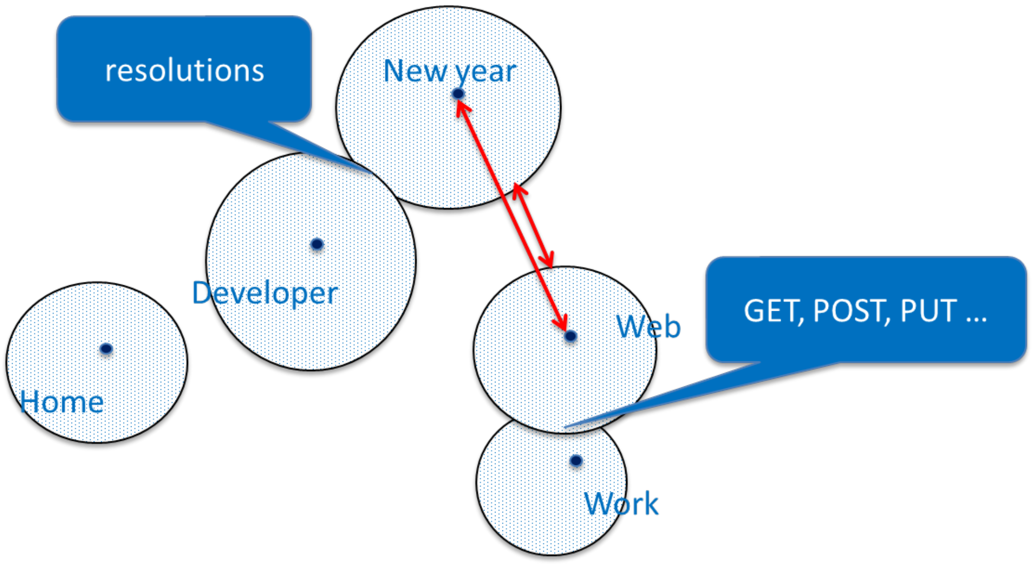

Figure 2: Expansion turns points into areas, reducing distance and making intersections

Here's one such place:

- HTTP methods are English words: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS ...

[12] http://www.developsense.com/presentations/2009-10-Noticing.pdf

- The developer is at work
- Workplace terminology includes the common English words get, post, put, options ...

So we're in, and I quickly tweeted a few gags back to the thread:

> *Q: Why was the web developer sent home early on new years?*
> *A: Was he sent home or did he desert his POST?*
> *A: Hopefully he was PUT in his place*
> *A: He's now considering his OPTIONS*

But that's an easy route. All of the jokes rely on the same underlying ambiguity. In software terms, all of the *failures* stem from the same underlying *fault*[13]: one problem manifesting with multiple symptoms in different places. For example, if your server has a bug in its file-writing library, saving from the client side might pop up an error dialog ("Terribly sorry, I couldn't save your file. Have a nice day"). But perhaps when the server tries to write to its log file it receives a fatal exception and crashes. The same fault provokes two different failures. Once you have established the fault, the value of additional bug reports about it generally decreases and may even become an overhead.

However, I had set a context in which those responses were delivered. I can reasonably assume that the original joke writer knows the methods, and knows that there are more than three of them, and that he can see that more gags could be generated that way, so I can potentially get a laugh by violating his expectation and not exploiting it further:

> *A: And so on ...*

I've employed a stopping heuristic here: the point at which the value returned from the effort put in reaches some threshold. But note that in the Twitter thread, others continued down that route, obviously continuing to see value where I saw none or insufficient. And that's my risk – there might be gold still to be found in the seam I've abandoned. Similarly, in software testing, when we stop exploring an area, there is always a risk that we've missed an important issue.

As well as value, we might consider other measures for our stopping strategies (each with their compromises) such as perceived risk, available time, relative priority of this project vs another, pressure from a stakeholder to do something else, and so on.

Back to the exploration, then. I decided to try another path through the "joke space" and so returned to the expansion of terms:

- JavaScript: common libraries such as node.js, jQuery, D3
- Node: sounds like "knowed" (he Node too much; it was because of what he Node), "no-ed" (The boss liked yes-men …)
- AJAX: sounds like "Hey Jacks", "Age Axe", "A Jacks"

I noticed that here I'd switched to sound relationships after thinking that "knowed" was a homophone of "node". I doubt that's going to be a generally useful testing approach but it is consistent with lateral thinking techniques of viewing concepts in unfamiliar ways. Decomposition is another technique, useful in both testing and joke-making, that I used that here too (e.g. breaking

[13] http://qahiccupps.blogspot.co.uk/2013/09/errors-by-any-other-name.html

"AJAX" into "A" and "Jacks"). And, just as I do when testing, I found my mind pursuing parallel lines not directly related to the main mission at one point:

- eleph-ant - the biggest insect in the world
- unpleas-ant - the worst insect to be around
- deodor-ant - the nicest smelling insect

But I pulled myself back to the task in hand …

- jQuery: sounds like "Jake weary"

I liked it in part because it includes a name – it could be the developer – and it's semantically related to the tiredness in the original joke. This latter point has no relevance to the joke in isolation, but in the context of the sequence of jokes it adds something. Just as a set of bugs of moderate interest in the same area might together constitute something important.[14]

With "jQuery"/"Jake weary" I have a potential surprise between the JavaScript library and a person being tired. In Brownell and Gardner's terminology, for a joke I need to find a coherent setup. From a testing perspective, I have a *theoretical* issue and I need to find a context in which it is a *practical* one.

Imagine again the server from the earlier example. As a tester, I can go onto the back end and fiddle around with the file system and perhaps by doing so I can make the server crash and lose customer data. This is an interesting result. It's a violation of a reasonable expectation – the server shouldn't crash and lose data. (It's a *surprise*.) If I can now work backwards from this to a context in which the user could provoke the same effect from the client side, without privileged access, then I'll have a context in which it's important to look at fixing. (The surprise will be *coherent*.)

In the joke domain this might map to a question like "can it be fitted into a sentence?"

- Jake said he was weary
- The boss saw that Jake was weary
- Jake's weary programming

I needed to get "Jake" adjacent to "weary" to get the homophone. At this point I am iterating ideas and evaluation to generate more ideas and hopefully find the context I need to make a joke. The goal here is to spur more ideas, hoping to find a useful one, or an interesting pattern.

Day-to-day at work this is the kind of thing I'll do with a potential issue; I can see it *could be* a problem but can I find a variant of it which *is* a problem? When working with others I've found that it pays to be very clear that I've shifted to a speculative mode of working[15] in this kind of situation.

Another thought …

---

[14] But see http://www.developsense.com/blog/2009/01/ideas-around-bug-clusters for thoughts on what it means to be "in the same area" and some potential perils of setting out to look for issues clustered around one you already know. This latter point might be related to the difference between the humour derived from a running joke (where somehow repetition and subtle variety across retellings over time enhance the effect) and that derived from pursuing one line of joke to exhaustion (where the humour soon wears thin).

[15] http://qahiccupps.blogspot.co.uk/2012/07/elephant-in-fume.html

- Was Jake weary?

It could work, but doesn't tie to anything else. I wondered whether it could be tied to a work concept, so more expansion …

- work: colleague, co-worker (idea: worker threads), desk, cubicle, machine, debugger, manager

... which generated another attempt ...

- The boss saw Jake weary

... which was better but still not a big laugh.

I've mentioned the relative rule already, but there's another important concept in testing that I'm implicitly also using in this process, the oracle[16].

In the construction of a joke, I am my own oracle. Does it make me laugh? If so, it might make someone else laugh. I am prepared to trust my own judgement in the right context (even as a tester, I am also a user of software, so I can have an opinion about how an application should work) but I also frequently ask colleagues for their opinion on my observations before filing a bug report.

The gag we're working on – for me – isn't ambiguous enough yet. The better laugh comes with more surprise but without losing coherence. And similar to the need/want trick we pulled earlier, I wonder whether we can introduce it with a verb here:

Q: Why was the web developer sent home early on new years?
A: The boss hated to see Jake weary

Notice that we have kind-of negated the previous attempt "The boss saw Jake weary". This can be a useful testing heuristic. Imagine the opposite (and also lack) of whatever you're looking at. Would that be better or worse? Why? How could that state arise?

Continuing the search, and looking for another angle of attack, I realised we still hadn't used the seasonal aspect of the original setup. So I generated some more links:

- resolution: new year resolution, image resolution, png, gif, jpg, lack of resolution, poor resolution,

Can we make a joke out of that lot? In **ARE YOUR LIGHTS ON?** Jerry Weinberg applies a set of heuristics (including a dictionary game) to a nursery rhyme to generate variants and arrives at a joke based on the ambiguity in "had":

Mary had a little lamb.
The event made medical history.

In my case, rote generation again threw up something that I thought looked interesting:

- his image resolution was poor.

[16] http://kaner.com/?p=190

It could work as a joke if the "image resolution" is also obviously a New Year resolution. Can we find a link?

- his image Resolution was poor

I capitalised "resolution" to make it more of an obvious point of ambiguity. In speech I might emphasise it – both of these are clues that one of the interpretations is much less likely than the other, in the context and so it might be hard for a listener to spot the potential surprise.

But I was struggling to make this work (well enough) in text so I quit ... and at that point had a last idea. I think of this as the *Columbo heuristic* where you say you're finished and hope to trigger that crucial "one more thing". Ding!

- he didn't care about resolutions
- he didn't care about image resolutions
- he didn't care about *his* resolutions
- he didn't care about his image resolutions

Now I'm trying to balance the ambiguities. Putting "image" in front of "resolutions" feels to me like biasing the interpretation towards graphics and away from the calendar while emphasising "his" seems to make the link to the new year strong enough.

To my testing mind now I'm doing something akin to establishing a reason why a user would perform some action: there is a plausible route through the application which includes picking up just the right data, or establishing just the right context, that the issue can be seen.

And so we have a joke:

> *Q: Why was the web developer sent home early on new years?*
> *A: Because he didn't care about his resolutions.*

In some ways I like this one the best because it manages to pull together the concepts in the setup – the web development, the time of year – and also make some distinct and plausible readings:

- He wasn't taking care with the images he was using.
- He'd said he would change in some way and didn't.

But I don't think it's the funniest of the bunch. And it reflects another reality of software testing that you've almost certainly encountered: reporting a bug that you're proud of, perhaps because you went the extra mile to discover it, used a novel approach, came up with something that no-one else had thought of … and then discovered that nobody who matters cares.

Really?

Really? is a great question. Perhaps you're thinking to yourself "Is there *really* this kind of connection between jokes and tests? I don't have a *joke plan*! Heh, heh, heh."

Well, congratulations, *you* may not have. But I bet you know someone who has what are effectively "joke cases", those lines they trot out whenever some trigger occurs. Everyone becomes used to them, bored of them even, and so they have increasingly little effect. Does that sound familiar?

Interestingly, it's possible to repurpose an old joke; and also test ideas, with application to a new environment, or person, or other contextual factor. A joke I know was funny when we had a king; can I make the same joke but about our new Prime Minister? I found a bug in a previous application using this idea; can I find one in this similar application by re-using it?

More generally, I'm not the first person to have thought about the analogue between investigative, exploratory work and humour. Hurley, Dennett and Adams, in **INSIDE JOKES**, have an intuition about it too:

> *we science-minded theorists keep finding deep parallels between humour and scientific investigation.*

They build a broad and deep psycholinguistic model of humour which includes the insight that (the cognitive mechanisms of) humour and investigation are substantially similar. In their theory, a significant difference between the two is the degree to which a *belief* (some element of a mental model) is *committed to*.

When investigating, you will often deliberately assume something is true and then look for evidence that it is not – this is a belief that you are *not* committed to and there is no humour in finding that it is false. When hearing a (well-crafted) joke you will be implicitly committed to a number of beliefs – cognitive heuristics will make assumptions on your behalf – and when one of them is violated, the humour comes.

They go on to say:

> *It is telling that we can often if not always devise some kind of context in which an incongruity turns into a humourous circumstance*

Essentially, once a problem has been discovered it's likely that a context can be concocted in which the key belief could be primed and then violated to give humour (the twin notions of surprise and coherence that we've seen already). The degree to which this is funny will depend (to a large extent) on how convincing that context is to the listener.

"OK," you might counter, "but if joking and testing align so well, why aren't we testers chuckling away all day long?" That's another good question. Violated expectation, new connections, exposed ambiguity, and so on are *necessary* for a joke but not sufficient. I would be astonished if you have never laughed on finding a bug; but it's likely to have been on a *meta-laugh*: "how could *that* have got into production?" or "why didn't I think of *that*?"

This is consistent with Hurley et al's ideas about beliefs: the chuckles in investigation often come when you realise that something you had considered axiomatic or very unlikely to be incorrect or just assumed unconsciously – and hence were committed to – is false.

So What?

Maybe you accept that it's possible to make a case for some kind of isomorphism between testing and joking by this point. Even if you do, there's another good testing question to ask: *so what?*

Jonathan Miller, in **Laughing Matters**, posits that jokes provide an opportunity to explore "category switching". What he means by this is that it permits us to take an item from its usual context and place it in another. He references a famous scene from The Gold Rush[17] where Charlie Chaplin eats a shoe. But shoes are not food[18] and Miller's point is that though you *could* eat a shoe we generally choose not to, although this is by convention and convention can be broken once it is *observed to be a convention* and not some immutable law.

In testing you will frequently be looking at functionality that's nominated for some particular use. And that's how people (intend to) use it. But what prevents us from using it another way? What happens if we do? The notion of functional fixedness[19] is very relevant here and I'm sure you've come across those "how would you test a pen?" or "what uses could you make of a brick?" exercises which encourage thinking outside of categorical boundaries. There are humorous contributions to this kind of thing: a brick is funny when you drop it on someone's foot; when *they* drop it on their *own* foot; when you smash a child's Lego house with a breeze block[20].

Athletes train for an event by doing drills and exercises that are *not* the event, but which share enough of some aspects of it to help build up strength, muscle memory and skills that will be useful *in* the event. I see and use joking as training[21] for (aspects of) testing such as expansion, factoring, finding ambiguity, making connections, maintaining multiple opposing viewpoints, exploring a search space, stopping heuristics, lateral thinking, idea generation, bug advocacy and creativity.

You may already practice your testing skills using games or katas or in workshops or peer review. Joking is not dissimilar, but you can practise it anywhere, anytime, without any tools, materials, other people or software. And this kind of exercise helps us to form intuition and build useful habits. There's a George Carlin quote[22] I like a lot on this (my emphasis):

> *the brain is a goal-seeking and problem-solving machine, and if you put into it the parameters of what it is you need or want or expect,* and you feed it*, it will do a lot of work without you even noticing.*

Quite apart from agreeing with the sentiment, it amuses me that here Carlin is exploiting the unconscious cognitive processes that humour itself exploits in order to make jokes that exploit that mechanism.

Crafting jokes can make you more alert to the recognition of nuance, the kind of subtleties that are required to balance the ambiguities and remain both surprising and coherent. This attunement, I

---

[17] https://www.youtube.com/watch?v=gY0DOnNK3Wg
[18] Apart from *cashews*, obviously, and maybe Dover sole or eel pie?
[19] http://qahiccupps.blogspot.co.uk/2013/03/unfix-this.html
[20] Perhaps an example of the relative rule, this one. Or maybe I'm just a bit sick.
[21] http://qahiccupps.blogspot.co.uk/2015/02/the-rule-of-three-and-me.html
[22] http://splitsider.com/2013/12/george-carlin-on-how-he-came-up-with-material-advice-to-young-comedians-and-his-influence/

find, makes my own writing more clear more of the time, first time. On the flip side, I am also alert to the possibility of ambiguity in the materials that I look at.

Becoming practiced at telling jokes can also help your reporting. As the comedian Frank Carson has it informally, "It's the way I tell 'em". I'd be surprised if we haven't all seen an issue we've reported be ignored and the same issue reported by others be accepted or, alternatively, some trivial piece of nothing reported by the right person getting very high priority.

Bug advocacy[23] is all about how you tell 'em: if you want the best chance of your report being acted upon learn how to deliver[24] by considering the content, the context, the time, the person you tell it to and their biases and the confidence with which it's delivered.

Summary

I find interesting parallels between testing – in fact, investigation and exploration more generally – and making jokes and I've identified areas where I think this is relevant to testing.

I've described some of the techniques I use to help find *surprise* and to increase *coherence* and talked a little about the necessity for good *delivery*. I've illustrated how those techniques can cross over, for me at least, and the kinds of benefits that I think can accrue from that, including breaking out of functional fixedness, becoming more sensitive to ambiguity, increasing lateral thinking ability and improving reports.

I was joking when I said at the top that your testing is a joke. But I'm not when I say that mine is.

Selected References

Lateral Thinking, E. de Bono, Penguin (2009)

Laughing Matters, J. Durant and J. Miller, Longman (1988)

Inside Jokes, M. Hurley, D. Dennett, R. Adams Jr., MIT Press (2013)

How I Escaped My Certain Fate, S. Lee, Faber & Faber (2010)

Are Your Lights On? G. Weinberg and D. Gause, Weinberg & Weinberg (2011)

Credits

The comments, suggestions, criticism and encouragement from the following people on various aspects and versions of this ebook, and the associated talk and EuroSTAR paper were much appreciated: Sneha Bhat, Michael Bolton, Laurent Bossavit, Graham Freeburn, Paul Gerrard, Paul Gilson, Adam Knight, James Lyndsay, Amy Phillips, Declan O'Riordan, Patrick Prill, Josh Raine, Ros Shufflebotham, Aleksandar Simic, Karo Stoltzenburg, Damian Synadinos, Ichiko Tanabe, Jason Trenouth, Sue Underwood, Jerry Weinberg, Peter Whalley.

[23] http://www.testingeducation.org/BBST/bugadvocacy/
[24] http://www.wikihow.com/Tell-a-Joke