Using Heuristic Test Oracles

By Michael Kelly

Date: Apr 28, 2006

Return to the article

When you see something that bugs you about an application, can you easily convince someone else that the defect really is a problem? Michael Kelly teaches a quick lesson in expressing yourself persuasively with a HICCUPP.

Quick! Test the application in Figure 1! No really, test it. It's Google Earth (Beta version 3.0.0762). It's okay, I'll wait for you. Just don't take more than an hour. I've got other people to share this example with.

Preguntas:
1) Sive para explicar el testing, pero solo para eso?
2) En qué casos nos puede ser útil realmente?

Figure 1 Very cool Google software.

How did you do? Did you find any bugs? I found the little beauty in Figure 2just a couple of minutes into my testing. It seems that if you search with large strings, Google Earth thinks it has a network error. What's even more interesting is that it actually changes your search criteria. I found that if I searched with a 5,000-character counterstring followed by a 1,000-character counterstring, Google Earth changed my search criteria back to the 5,000-character counterstring. It continued to do this for several different values.

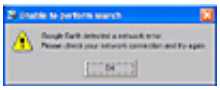

Figure 2 Searching with large strings led to this error message.

Is this a good bug? Probably not—there's a low likelihood that someone will actually have a valid reason to search with 5,000 characters. Is this a bug that should be fixed? Eventually—most likely there are more important bugs to fix (like the funny white lines when rendering). However, bugs like this are like blood in the water. They tell me that the developer may not have the control over the application that he thinks he has.

What does all this have to do with heuristic test oracles? Suppose you ask me, "How do you know the behavior you found above is a bug? You don't have any requirements telling you what the software should do." To that, I might answer, "Uh, well...it just didn't seem right to me." But that response wouldn't win many supporters, would it?

Instead, I need to say something like, "I believe it's a bug because the message states that there's a network problem, and yet searches with smaller values still work. The search feature behaves inconsistently given the same network conditions. In addition, the search field changes my search text after I get the error message, but it doesn't do that *before* I get the error message. That's also inconsistent behavior. Finally, I truly don't believe that there's a network error, which means that Google Earth is reporting an inaccurate error message. I don't believe Google really wants to report inaccurate error messages. That makes them look bad. That's not good for their company image."

The rest of this article is intended to help you develop a vocabulary for explaining why you feel that something you find that looks wrong is in fact a bug. Rather than being forced to rely solely on incomplete requirements ("Where's the requirement that says we should spell everything correctly?") or having to put your reputation on the line ("I think it's a bug, and you should fix it because I'm a good tester"), this article offers some simple heuristic test oracles you can use to explain why you think something is a problem.

Enter the HICCUPP Heuristic

Last year, while I was working with James Bach, he had me test some small applications. When I would find a problem, James would ask me to explain why I thought that what I found was a problem. I quickly discovered that I had no vocabulary for why I thought that something was a problem. There were no requirements documents around, so I couldn't point to something and say, "Look there. It's a problem because that says so!" Instead, I just looked at him and said, "Well, it seems like it should work like *this*, not like *that*."

Obviously, that wasn't good enough. James is fairly good at role playing, and I quickly got everything from a defensive developer ("Well, I don't care if you're a good *tester*—I'm a good *developer*, so it's not a bug!") to a dismissive project manager ("Well, that's just *your* opinion"). It seems that most people don't react well when they suspect you're being subjective.

James gave me the following handy mnemonic for test oracles (the principle or mechanism by which we recognize a problem): HICCUPP. Here's what the letters stand for:

- **H**istory
- **I**mage
- **C**omparable Product
- **C**laims
- **U**ser Expectation
- **P**roduct
- **P**urpose

Let's consider how you address these oracles:

- *Inconsistent with History.* A product should be consistent with past versions (or history). History can include previous versions, patches, claims, etc. If something has changed, and no one told you it was supposed to change, then you might have found a problem.
- *Inconsistent with Image.* Most companies want to have a good image in the marketplace. Therefore, their software needs to look professional and be consistent with accepted standards. If a product is inconsistent with the desired image, what you're saying is this: "We'll look silly (or unprofessional) if we release this software to market."
- *Inconsistent with Comparable Product.* You're letting another product serve as your oracle for this test. As long as the comparable product really *is* comparable, and you want your product to be an alternative to that product, or you want to get the users that that product has, then this oracle can be very compelling.
- *Inconsistent with Claims.* A "claim" can be anything that someone in your company says about the product. If something is inconsistent with claims, it's inconsistent with the product's stated requirements, help, marketing material, or just something that a project stakeholder said in the hallway.
- *Inconsistent with User Expectations.* This product doesn't do something that a reasonable user of this product would expect it to do, or doesn't perform a task in a way that the user would expect. Using this oracle means that you have some idea of who the user is and some indication of what he or she expects.
- *Inconsistent Within the Product.* Something behaves in one way in one part of the product, but in a different way in another part of the product. The change could be related to terminology, look and feel, functionality, or feature set. All you're doing is pointing out where the product is inconsistent with itself. These are often compelling bugs.
- *Inconsistent with Purpose.* In my mind, this is the most compelling of the oracles. It states that the behavior you found is contradictory to what a user would want to do with this software. You might be talking about the purpose of a feature, for example: "Look, I can enter a negative value for headers in this Microsoft Word dialog box." That wouldn't really be consistent with the purpose of headers and footers, would it? This oracle is often used in conjunction with *Inconsistent with Claims* or *Inconsistent with User Expectations* because those oracles also tend to address the purpose of the software.

Crossover Among Oracles

One thing you'll notice quite quickly is that when a product violates one oracle, it often violates others. That's because most inconsistencies conflict with user expectations, and if we think something's inconsistent we think that's bad for the company's image. That fact points to the relative strength of each of these oracles:

1. I've found the oracles *Inconsistent with Claims*, *Inconsistent with Comparable Product*, and *Inconsistent with History* to be the most effective in getting bugs fixed.
2. After those, *Inconsistent Within Product* and *Inconsistent with Purpose* tend to influence developers, but rarely managers.
3. Finally, *Inconsistent with Image* and *Inconsistent with User Expectations* tend not to get much notice in the defect-resolution world. While meaningful to me (both expert user and marketing guru), they tend to be considered oddly subjective by the people who prioritize my defects. Translation: I don't normally file defects based on these two oracles alone. I use them like frosting on an already buggy donut.

HICCUPP Applied

Let's go back to Google Earth and run some more tests, in which we apply some of these oracles. I spent another five minutes with the application and found the following problems. I'll share with you how I found each problem, and why I think it's a problem (using our test oracles).

Negative Values for Altitude

A quick blink test allowed me to find a negative value for eye altitude. It would seem that if you zoom all the way out, tilt all the way down, and then just hold down the Zoom In button, you can get Google Earth to show negative altitudes (see Figure 3).

Figure 3 Negative value for eye altitude.

An odd little combinatoric problem (I can't get this error unless I start all the way out and I have the tilt all the way down), this result illustrates the following behaviors:

- *Inconsistent Within the Product.* I can't zoom to a negative altitude if I don't apply a tilt.
- *Inconsistent with Purpose.* The purpose of zooming and tilting isn't to get negative altitudes—it's to get a better view of the landscape.
- *Inconsistent with User Expectations.* I certainly didn't expect to see a negative value.

Rendering Lines When Selecting Layers

Another simple test is to select multiple layers and then simply navigate around the globe, zooming in and out when it takes your fancy. When I select more than one or two layers, a series of white lines becomes visible when rendering. If you select many layers, the problem becomes very visible (see Figure 4).

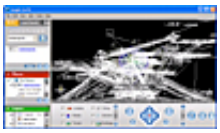

Figure 4 Rendering with multiple layers selected.

This result demonstrates a couple of problems:

- *Inconsistent with Purpose.* The lines obscure my ability to see the actual map.
- *Inconsistent with Image*. This is a fairly basic problem that I discovered the first time I ever used Google Earth (months ago). It looks bad, and since I like Google's software so much, I expect more from Google.

Different Travel Times for Directions

While trying to think of an error for an inconsistency with a similar product, I thought I might try comparing Google Earth with Google Local. Sure enough, the very first test I ran showed a difference. I ran a quick search for directions to my dad's house and found that the suggested drive times differed by 24 minutes with the same distance in miles and the same suggested route. Figure 5 shows the distance in Google Local; compare the distance in Google Earth shown in Figure 6. This test illustrates two problems:

- *Inconsistent with Product.* Both Google products should get the same result.
- *Inconsistent with Image.* Google makes both products; their inconsistent behaviors aren't good for Google's image.

Figure 5 Distance in Google Local.

Figure 6 Distance in Google Earth.

An interesting bit of follow-up on this bug found that if you click the Printable View link in Google Earth, it opens a browser window with the Google Local directions, which displays the difference in time. So there go any arguments that Google Local might not be a comparable product (for this feature, at least)!

Inaccurate Documentation

This leads us to our next find. The Google Earth help file (beta, last updated January 18, 2006) claims, "The Google Maps web page appears on the bottom half of the screen with the directions highlighted on the page along with directions" when you click the Printable View link. It even shows an image of the two applications working together (see Figure 7).

Figure 7 Image taken from Google Earth documentation.

Rather than appearing at the bottom of the page, when I click the link the web page opens in my Firefox web browser. This is *Inconsistent with Claims*. On the other hand, I don't feel that it's *Inconsistent with User Expectations* or *Inconsistent with Image* because the functionality as it worked met my expectations as a user, and I think it's good for Google's image to show that their application works with other products such as Firefox.

Okay, enough beating up on Google Earth. It's in beta, after all! Sadly, I didn't get a chance to show you an *Inconsistent with History* example, but that's what happens when your sample application is in beta. However, I'm sure you get the idea.

You Take It from Here

Remember that the HICCUPP test oracles are just heuristics. A heuristic is a solution to a problem that works most of the time. Translation: The HICCUPP heuristics are fallible! Relying solely on the HICCUPP test oracles is not something I would recommend. If you have requirements available, use them. If you have actual users available, quote them. Use these oracles to either support your existing data for a bug or to explain other

aspects of your bug that may not be apparent if you only look at requirements.

Practicing the HICCUPP heuristic is easy for most testers to do. Every time you log a new defect, before you click Send see whether you can answer the following questions:

- Is this defect inconsistent with the product history?
- Is this defect inconsistent with the image our company (or project team) is attempting to portray?
- Is this defect inconsistent with a comparable product?
- Is this defect inconsistent with claims made about the product?
- Is this defect inconsistent with user expectations about the product?
- Does this defect show an inconsistency within the product?
- Is this defect inconsistent with the purpose of the product?

By going through this process for the types of defects you currently find, you'll train your mind to find defects based on these criteria in the future. It's a quick and powerful way to improve the way you think about your testing.
