

# Color Coding Theory

How lunch with a Monopoly board  
turned into an ongoing dialog and  
testing theory with a University  
Professor from Canada.

The green letters  
stood out didn't  
they? See – it's  
already working

A work in progress by McGill University professor  
Robert Sabourin and Ben Brodsky

# What am I talking about?

Color coding theory is a method of test creation which aims at giving value by shaping, finding relevance and organizing tests based on overarching and broad testing concepts.

By categorizing tests at a high level, test concepts designated by color, we can quickly and easily focus our aim and maximize reliable test coverage in a collaborative method to expose risks in the test plan.

# There are more possible tests than particles of matter in the known Universe

Professor Edsger Dijkstra was reported as saying,

*“program testing may convincingly demonstrate the presence of bugs, but can never demonstrate their absence.”*<sup>[1]</sup>

Even the smallest and limited piece of software we know that the testing possibilities are quite unfathomable. Certainly many more tests exist on even a small piece of code than there are particles of matter in the known Universe.

As much as we may wish to try, it’s absolutely impossible to test everything. I’ll take a risk and speak for most of us by saying we could find a better and more focused way of spending our day than attempting the impossible.

So what do we do? We want to have that warm and fuzzy feeling in our bellies that tells us we did our due diligence.

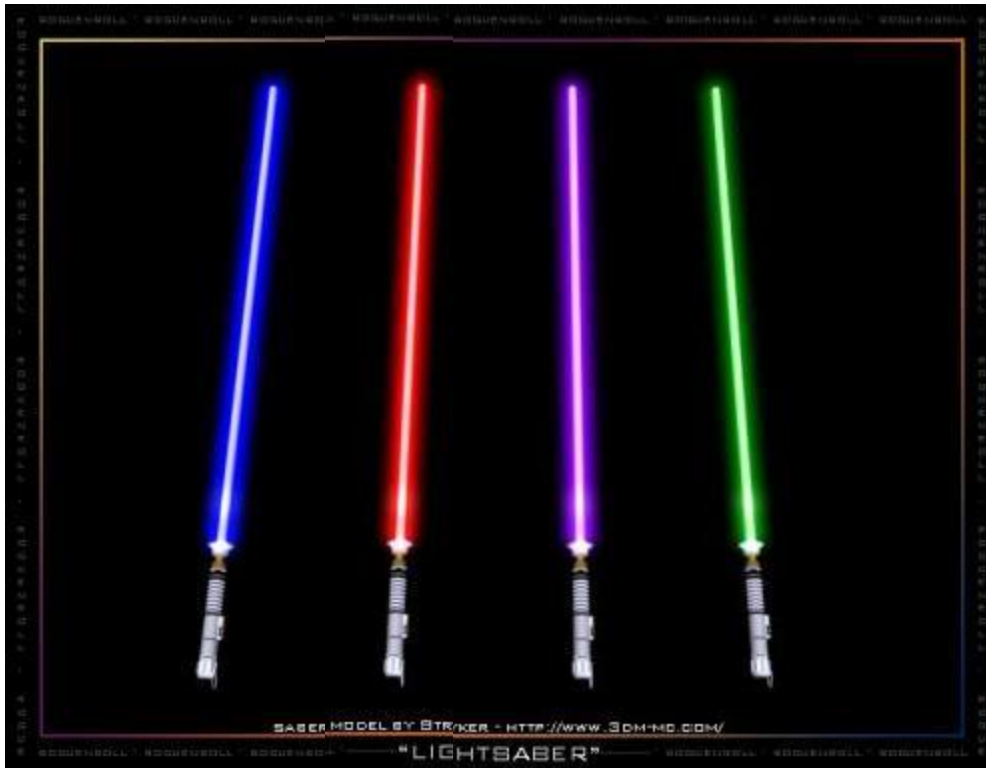
How do we get there?

[1] *EWD1036: On the Cruelty of really teaching computing science.* Professor Edsger Dijkstra; Department of Computer Sciences, University of Texas at Austin. December 2nd 1988

<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html>

# Lets look around us for an answer

Popular culture uses color coding to show good vs. evil



Blue signifies a hero

Red signifies a vilian

Green signifies a little hero

Purple = Samuel L. Jackson

# Even fake militaries use color coding. Take Star Trek for example

*Star Trek* uses color to distinguish the department colors:

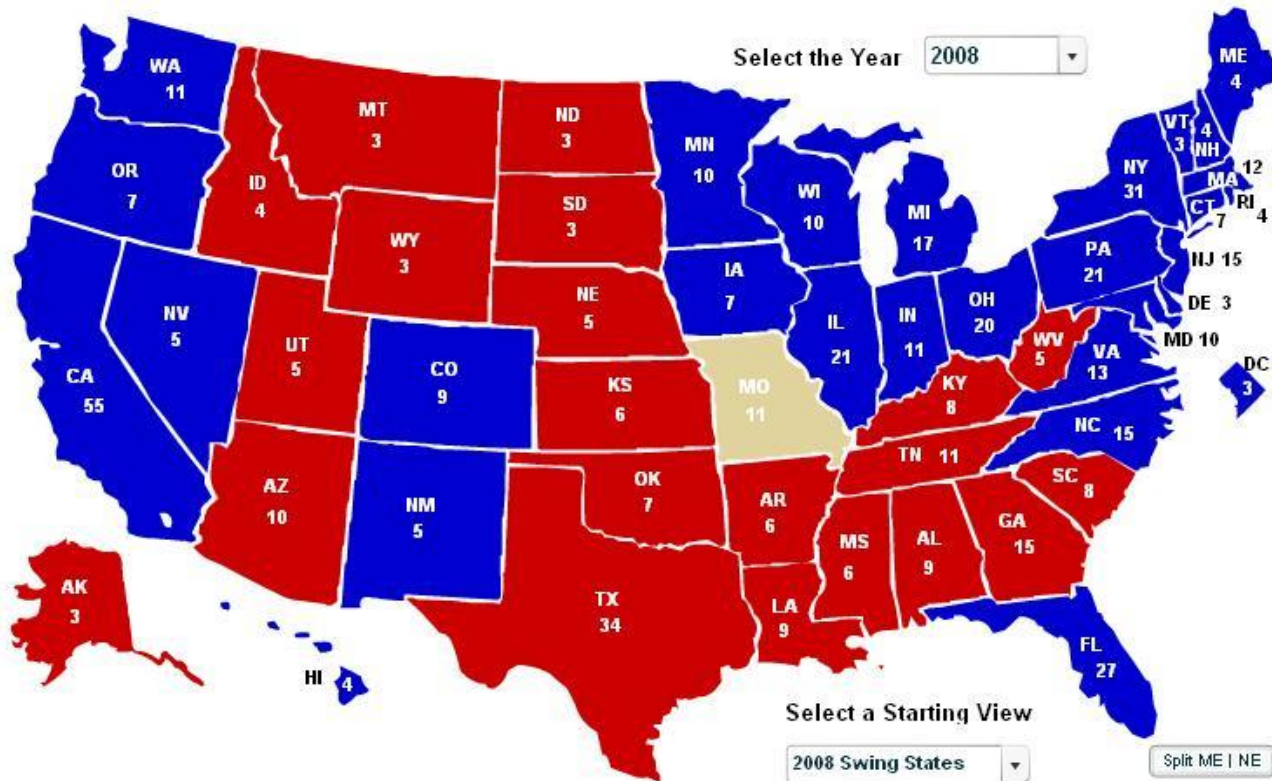


Command and  
Helm personnel  
wear gold shirts,

Engineering and  
security personnel  
wear red

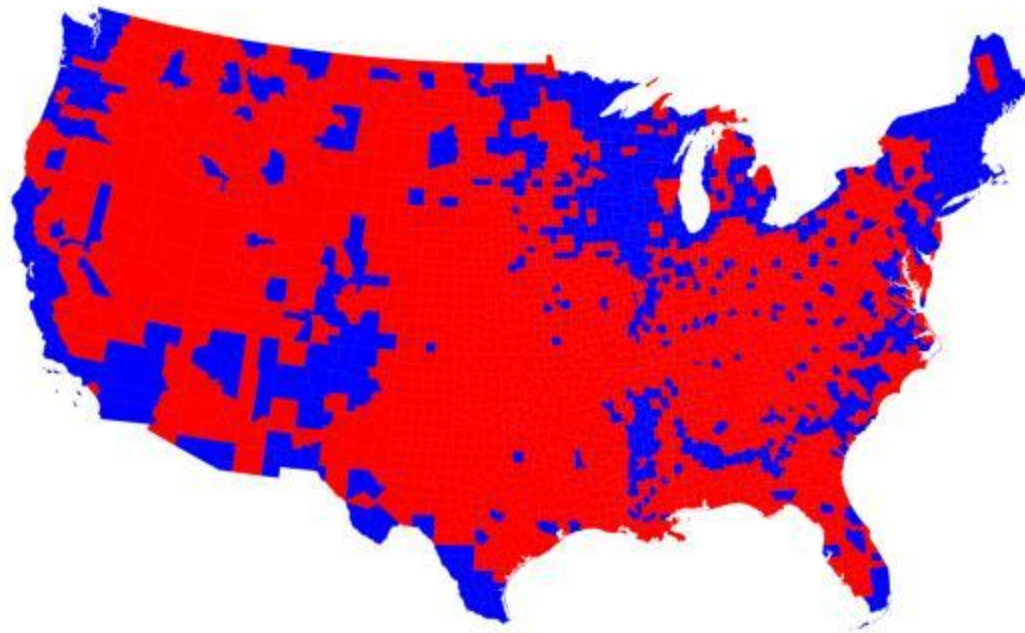
Science and  
Medical personnel  
wear blue

# How we break down our information counts



Examining the United States broken down by the states won by the Democrats vs. Republicans, shows us one view of the results

Viewing the same color coded data  
in a different format, our view can  
look different.



Examining  
the United  
States broken  
down by the  
Counties won  
by the  
Democrats  
vs. the  
Republicans,  
shows us  
another view  
of the results

# Lets get back to your question of “How do we get there?”

One response, among many, is that we must be smarter than the construct.

We must priorities our efforts with help from others, i.e., product champions, managers, and developers, etc, to streamline our testing.

The main road our color coding theory dances down is the road of **open communication** (corny, I know; albeit true).

When we first sit down and begin to organize our thoughts, it would be wonderful to already have an agreed upon grouping of tests, or a test bank of sorts from which we could pick and choose from to best design our **first draft** of our test plan.

# Creating a test bank

When a phenomenal testing notion strikes us, the most constructive thing I know to do is to catalog it. Save it. Keep it safe for when we need it down the road.

However if we are rolling along with this test catalog analogy, we need to take another step and categorize our test ideas.

If we open a computer hardware catalog for example, within the first few pages we should find some sort of glossary: Monitors: pg. 4-6, Memory: pg. 7-9 etc. What they have done is grouped specific objects within larger constructs. This is something we can and should do within testing.

We can group tests based on larger constructs. With this mindset in place we can easily locate just the right test for our product.

# Test Case Catalog

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	Test Case #	Priority	Major Area	Sub Area	Section	Test Case Title	Steps (not always req.)	Expected Results	P/F/B	VS#	Last run	Notes		
1		1	Windows	General	Basic Functionality	Overall	Verify all possible entry points to the search page (links from other pages, bookmarks, returning from results/interim, copy/paste url, refresh etc.)							
2		2	Windows	General	GUI	Overall	Verify the default field data and initial states of the search page.							
3		2	Windows	General	GUI	Overall	Enter the search page from different points and verify the default state of all the controls (Textbox data, collapse controls, drop down boxes, check boxes, default courts, default billing code)							
4		1	Windows	General	GUI	Overall	Verify all controls in window laid-out in a clear and pleasant manner (controls aligned, appropriate labels used, adequate spacing, etc.)							
5		1	Windows	General	GUI	Overall	Verify that the cursor is positioned in the first input field or control when the screen is opened							
6		2	Windows	General	GUI	Overall	Verify that if an entry is required on a field or such, that it is marked as such and that there is a Key on the page noting Required							
7		2	Windows	General	GUI	Overall	Verify all the text on the window for spelling, tense and grammar							
8		2	Windows	General	GUI	Overall	Verify the same font style(s) (typeface, size, formatting, etc.) are used consistently throughout the application							
9		2	Windows	General	GUI	Overall	Verify the same color scheme(s) are used consistently throughout the application.							
10														

# What has Robert Sabourin done for us lately?

Robert has helped us here. He has come up with some overarching categories which we can use to organize our test plan. What's more, he has color coded them.

Why is this important? It does a few things – it helps us quickly identify if we are lacking coverage in a certain area very quickly.

It also helps us change mindsets more easily when creating tests.

# Our color key exposed

- **Capabilities: Green:** (Often the beginning tests) Does the app do what it's supposed to do.
- **Failure Modes: Red:** ("What if" questions) Challenge the expected outcome.
- **Inputs/outcomes: Purple:** (Boundaries tests) push the limits on all input fields with string lengths (min/max)
- **White box testing White:** Reviewing design code and data schemas. Exercising the decisions and paths through the program.
- **Quality Factors: Yellow:** Performance, scalability, usability
- **Usage Scenarios: Blue:** Extremely high value tests: Ask not what the app does for the user, but what the user does for the app. View these from the perspective of the specific users. Can the user do their job with the application?
- **Creative Testing: Gold:** Soap opera testing. Do what the user would normally do, but much more dramatically. What if the user goes through things in the wrong order?

# Overlap?

Now you may have noticed that there are many tests that could overlap and touch more than one category. To this I say bravo to you for noticing.

Often the most valuable tests are those that dip into many categories.

So far so good – we now have a bank of tests or a test catalog from which we can shop from. Let's pick one from each category:





What we have just done is begun our test plan. As I'm sure you can imagine, coming up with this type of list is much easier and more precise if we have met and discussed the application with the development team, or product managers.

After we have exhausted ourselves coming up with our first draft of a test plan, the next step would be to gain some sense of organization and perspective based on priority. This is best done with the help of the product champion, or PM. They should have the best idea about what's most important and what can affect the business.

For example, they might not want any tests done on Non-Admin accounts. If this is the case, we are lucky because it frees up our time to focus on other items; tests we know will be better received.

# What's next?

A theory unfinished is a beautiful thing.

What do you think I have here. Could this be more dramatic or could it be more valuable for your team?