Investment Modeling
a Software Tester's Perspective

Cem Kaner, J.D., Ph.D.

Executive Vice-President, Association for Software Testing

Professor of Software Engineering, Florida Institute of Technology

For further discussion & debate on these ideas, come to my track talk and keynote at CAST 2010, Grand Rapids, August 2-4.

These notes are partially based on research that was supported by NSF Grant CCLI-0717613 "Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Every few years, someone gets the bright idea that what our field really needs is an emphasis on automated regression testing with tests that are focused specifically on written customer requirements, ideally developed as the requirements come in.

These days, I hear this tired, heavyweight approach marketed as "agile, lean, automated, acceptance-test-driven development." Huh? I thought this was what TDD (within XP) was trying to replace.

Oh well. To quote Peter Pan (and Battlestar Galactica),

"All of this has happened before, and it will all happen again."

I want to remind you of a **different** recurring theme involving test automation: using technology to help us learn things about the quality of the software that we don't already know, running new tests all the time so that we keep learning throughout testing.

The latest buzzphrase for this is "automated exploratory testing."

More specifically, I'm going to talk about applying test techniques to the evaluation of the models we build into software, to help us decide whether we are building the right thing, to address validation and accreditation of software instead of burying our heads in verification. (You can use automated exploratory techniques for high-powered verification too, but we won't have time for that in this presentation.)

"Trailing stops saved me a pantload"

Some people wonder whether I am talking about **software** testing.

http://www.youtube.com/watch?v=mDm6iKH38C0

Trailing stops are a feature of the E-Trade software and of most other trading software.

- Buy stock at $10
- Set a 10% trailing stop
- If stock drops 10% (to $9), automatically sell
- If stock rises 10% (to 11), then drops 10% (to $9.90) sell

This is automated trading.

Well over 50% of trades in US equity exchanges are automated.

Trillions of dollars are at risk when this software goes wrong.

Overview

1. Testers provide empirical research services, exposing quality-related information to our clients.
2. We play a major role in determining:
   - How valuable our research is for our clients
   - How much our clients will value our work
3. We can increase our value, if we can:
   - Find problems that have greater impact on the business, or
   - Use technology to find problems that are hard to find by hand
4. It's hard to reach these in software testing courses and books:
   - Deeper testing requires product knowledge. It can take a long teaching time to build enough product insight for a student tester to understand tests at that level
   - Exploratory test automation architectures call for deeper levels of technical sophistication than we can reach in most testing courses.

Overview - 2

- A year ago, I realized that remarkably many working professionals (and American university students) had some familiarity with the stock market.
- This presentation is the result of almost two year's work (so far) gaining an understanding of this domain and of how it might be used as a teaching foundation for complex issues in testing and high-reliability programming

Software testing

- is an empirical
- technical
- investigation
- conducted to provide stakeholders
- with information
- about the quality
- of the product or service under test

We design and run tests in order to gain useful information about the product's quality.

Empirical? -- All tests are experiments.
Information? -- Reduction of uncertainty. Read Karl Popper (Conjectures & Refutations) on the goals of experimentation

Commodity-Level Software Testing

You are a commodity if:

- your client perceives you as equivalent to the other members of your class

Commodity testers:

- have standardized skills / knowledge
- are easily replaced
- are cheaply outsourced
- add relatively little to the project

Commodities

There are green bananas
and ripe bananas
and rotten bananas
and big bananas
and little bananas.
But by and large,
a banana is a banana.

Commodity testers have little on-the-job control over their pay, status, job security, opportunity for professional growth or the focus of their work.

Typical Testing Tasks

Analyze product & its risks

- benefits & features
- risks in use
- market expectations
- interaction with external S/W
- diversity / stability of platforms
- extent of prior testing
- assess source code

Develop testing strategy

- pick key techniques
- prioritize testing foci

Design tests

- select key test ideas
- create tests for each idea

Run test first time (often by hand)

Evaluate results

- Troubleshoot failures
- Report failures

Manage test environment

- set up test lab
- select / use hardware/software configurations
- manage test tools

Keep archival records

- what tests have we run
- trace tests back to specs

If we create regression tests:

- Capture or code steps once test passes
- Save “good” result
- Document test / file
- Execute the test
  - – Evaluate result
    - ° Report failure or
    - ° Maintain test case

This contrasts the variety of tasks commonly done in testing with the narrow reach of UI-level regression automation. This list is illustrative, not exhaustive.

Automating system-level testing tasks

No testing tool covers this entire range of tasks

In automated regression testing:

- we automate the test execution, and a simple comparison of expected and obtained results
- we don't automate the design or implementation of the test or the assessment of the mismatch of results (when there is one) or the maintenance (which is often VERY expensive).
- So, the key design question is, where do we need the most assistance?

"GUI-level automated system testing" doesn't mean **automated testing**

"GUI-level automated system testing" means **computer-assisted testing**

GUI-Level Regression Testing: Commodity-Level Test Automation

- addresses a narrow subset of the universe of testing tasks
- re-use existing tests
  - a collection of tests that have one thing in common: the program has passed all of them
  - provide little new information about the product under test
  - tests are rarely revised to become harsher as the product gets more stable, so the suite is either too harsh for early testing or too simplistic / unrealistic for later testing
  - tests often address issues (e.g. boundary tests) that would be cheaper and better tested at unit level
- enormous maintenance costs
  - several basic frameworks for reducing GUI regression maintenance are well understood

Regression testing

- We do regression testing to check whether problems that the previous round of testing would have exposed have come into the product in this build.
  - Sometimes we are testing to confirm old bugs stay fixed, but ***most*** of that risk has been mitigated with modern configuration management practices
- We are NOT testing to confirm that the program "still works correctly"
  - It is impossible to completely test the program, so
    - we never know that it "works correctly"
    - we only know that we didn't find bugs with our previous tests

Regression testing

- A regression test series:
  - has relatively few tests
    - tests tied to stories, use cases, or specification paragraphs can be useful but there are not many of them. They do not fully explore the risks of the product.
  - every test is lovingly handcrafted (or should be) because we need to maximize the value of each test

Regression testing

- **The decision to automate a regression test is a matter of economics, not principle.**
  - It is profitable to automate a test (including paying the maintenance costs as the program evolves) if you would run the manual test so many times that the net cost of automation is less than manual execution.
  - Many manual tests are not suitable for regression automation because they provide information that we don't need to collect repeatedly
  - Few tests are worth running on every build.

Cost/benefit the system regression tests

COSTS

- Maintenance of UI / system-level tests is not free
- In practice, we have to do maintenance -- often involving a rewrite of the entire test -- many times.
- We must revise the test, whenever
  - the programmers change the design of the program, even in relatively minor ways.
  - we discover an inconsistency between the program and the test (and the program is correct)
  - we discover the problem is obsolescence of the test
  - we want to integrate new data or new conditions

Cost/benefit the system regression tests

BENEFITS?

- What information will we obtain from re-use of this test?
- What is the value of that information?
- How much does it cost to automate the test the first time?
- How much maintenance cost for the test over a period of time?
- How much inertia does the maintenance create for the project?
- How much support for rapid feedback does the test suite provide for the project?

In terms of information value, many tests that offered new data and insights long ago, are now just a bunch of tired old tests in a convenient-to-reuse heap.

The concept of inertia

INERTIA: The resistance to change that we build into a project.

The less inertia we build into a project, the more responsive the development group can be to stakeholder requests for change (design changes and bug fixes).

- **Process-induced inertia**. For example, under our development process, if there is going to be a change, we might have to:
  - rewrite the specification
  - rewrite the related tests (and redocument them)
  - rerun a bunch of regression tests
- Reduction of inertia is usually seen as a core objective of agile development.

We'll come back to the automation

But first, let's look at an example of testing that is central to business needs.

Let's consider:

Automate-what?

before discussing

Automate-how?

One of the motivators for me to study this area was the challenge of helping my students find work.

Finance has grown to 7% of the US economy (up from about 3%) while other sectors have been shrinking.

Unemployment For Young Americans Jumps To 52.2 Percent

First Posted: 09-27-09 12:47 PM | Updated: 09-27-09 03:32 PM

WHAT'S YOUR REACTION? Custom Search Search HuffPost

Amazing Inspiring Funny Scary Hot Crazy Important Weird

Read More: Economy, Jobless Rate, Jobs, Unemployment Rates, Young Americans Unemployment Rate, Business News

Buzz up! Be the First to Submit This Story to Digg

***New York Post*:**

The unemployment rate for young Americans has exploded to 52.2 percent -- a post-World War II high, according to the Labor Dept. -- meaning millions of Americans are staring at the likelihood that their lifetime earning potential will be diminished and, combined with the predicted slow economic recovery, their transition into productive members of society could be put on hold for an extended period of time.

Read the whole story: *New York Post*

Reports like this have been catching my attention:

Goldman Sachs compensation: $1 mn per employee in sight

4 Jul 2009, 2105 hrs IST, REUTERS,
http://economictimes.indiatimes.com/articleshow/4777281.cms?prtpage=1

NEW YORK: The average Goldman Sachs Group Inc employee is within striking distance of $1 million in compensation and benefits this year, just nine months after the bank received a $10 billion US government bailout.
The figure will likely fuel criticism of the politically connected bank, especially amid the widening recession and rising unemployment. In addition to the bailout, Wall Street's biggest surviving securities firm also benefited from several other government schemes during the depths of last year's financial crisis.

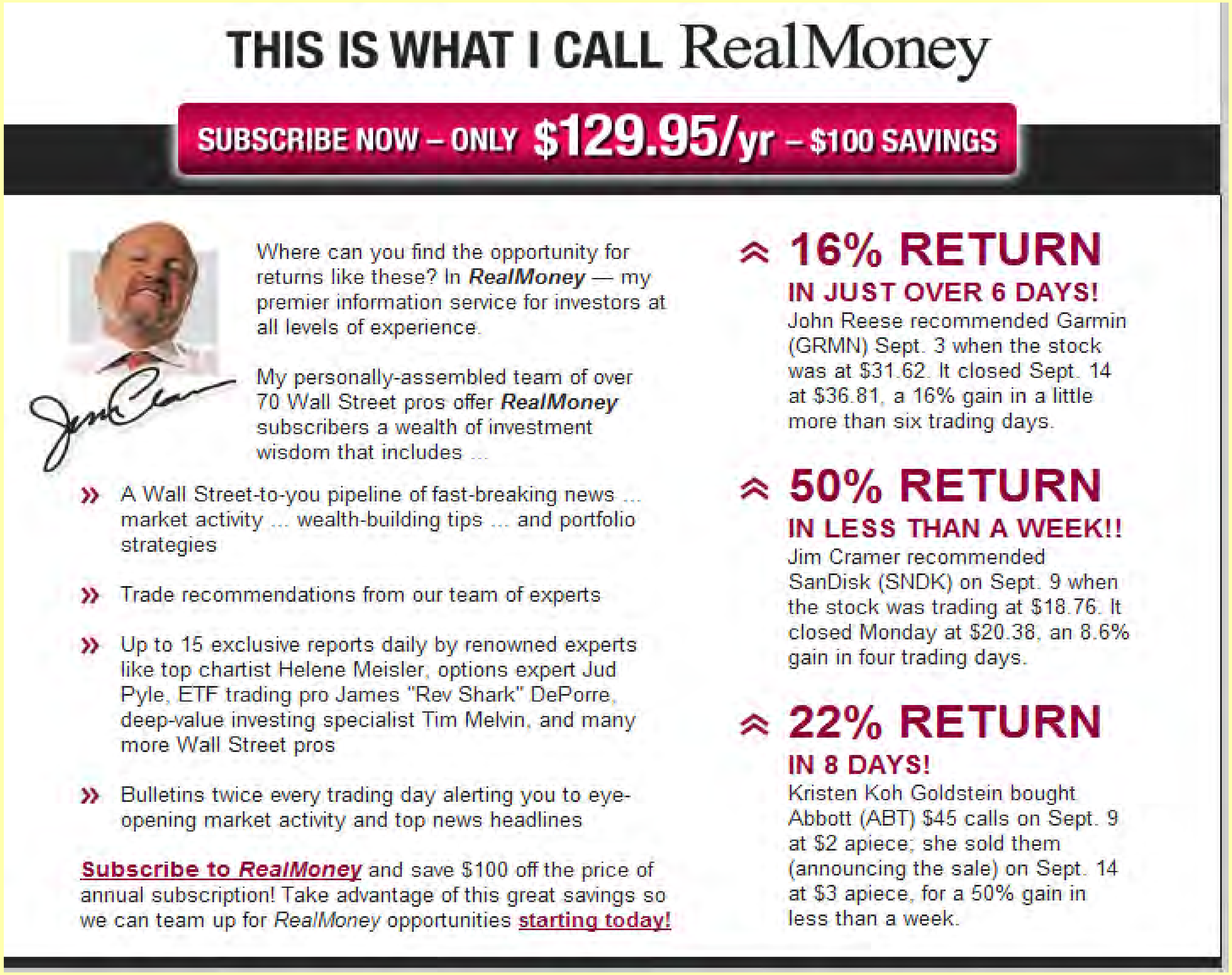
Goldman on Tuesday said money set aside for pay surged 75 percent in the second quarter. Compensation and benefits costs were $6.65 billion, up 47 percent from the equivalent quarter in 2008.

Given a 16 percent reduction in staff from last year, to 29,400, the bank set aside an average $226,156 per employee in the second quarter, up from $129,200 in a year ago. If the quarterly figure is annualized, it comes to **$904,624 per employee**.

A Critical Distinction

- People who create (or use) (or improve) technology to model the market are called "**quants.**"
  - **They are central to the business**
  - **The business bases its decisions on their work**
- People who write (or test) code are called "technicians"
  - Even if they are doing implementation for the quants
  - Their focus is on the generic tasks of development
  - Instead of the specialized tasks of the business
  - They are cheap, expendable, and outsourceable (but certifiable)

Non-professional investors often rely on advisors, and hope to beat the market by following the advice



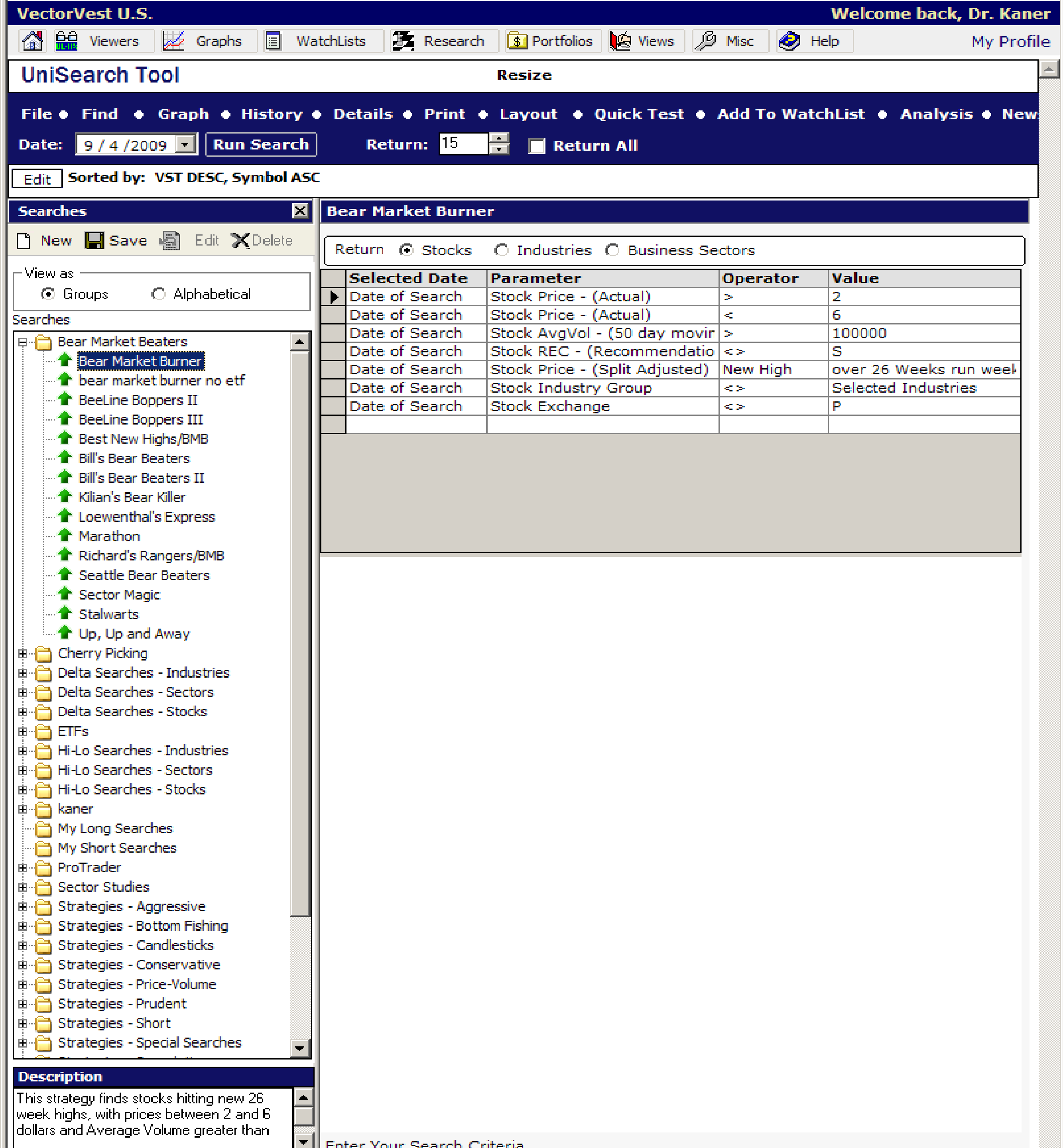
http://www.cnbc.com/id/18724672/
http://www.thestreet.com/tsc/emails/2009/rm_email_test_091909.html

Other individual investors rely on tools…

(NOTE: I use VectorVest in several examples because I liked it enough to research it more carefully than its competitors.

Despite my critical comments, you should understand that this product offers significant benefits, especially in the accessibility of its highly detailed historical fundamentals data.)

http://www.youtube.com/watch?v=lb_h_mwKk-o
http://www.youtube.com/watch?v=Whq4uQl2lYI&NR=1
http://www.vectorvest.com/freemovies/demo/vectorvestproducttour/vectorvestproducttour.html



VectorVest Strategies

There are about 250 of these, tailored for different expectations about market performance.

Strategy of the Week!

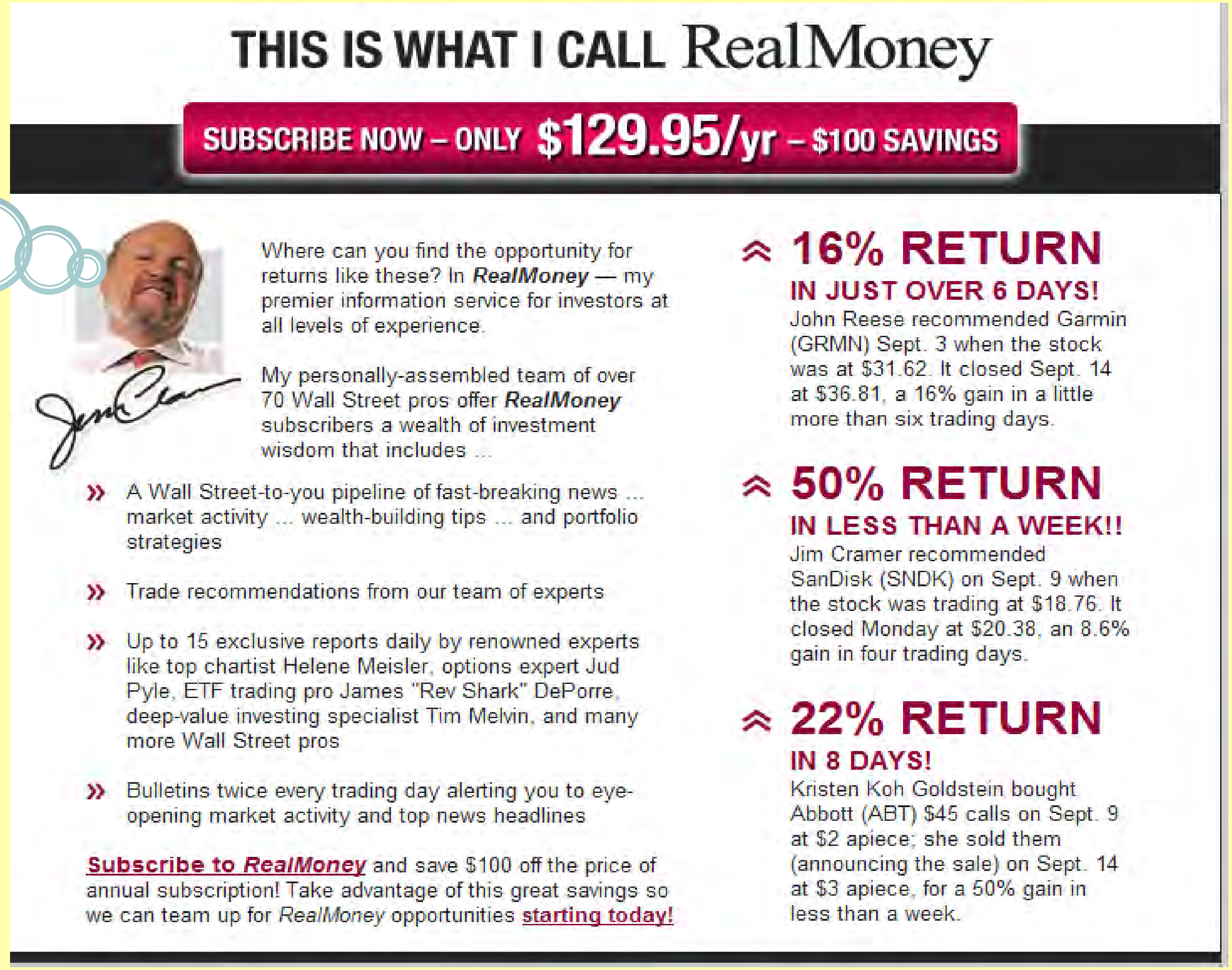
Suppose we sell a trading system with 250 strategies, and analyze the population every week to see which ones performed well.

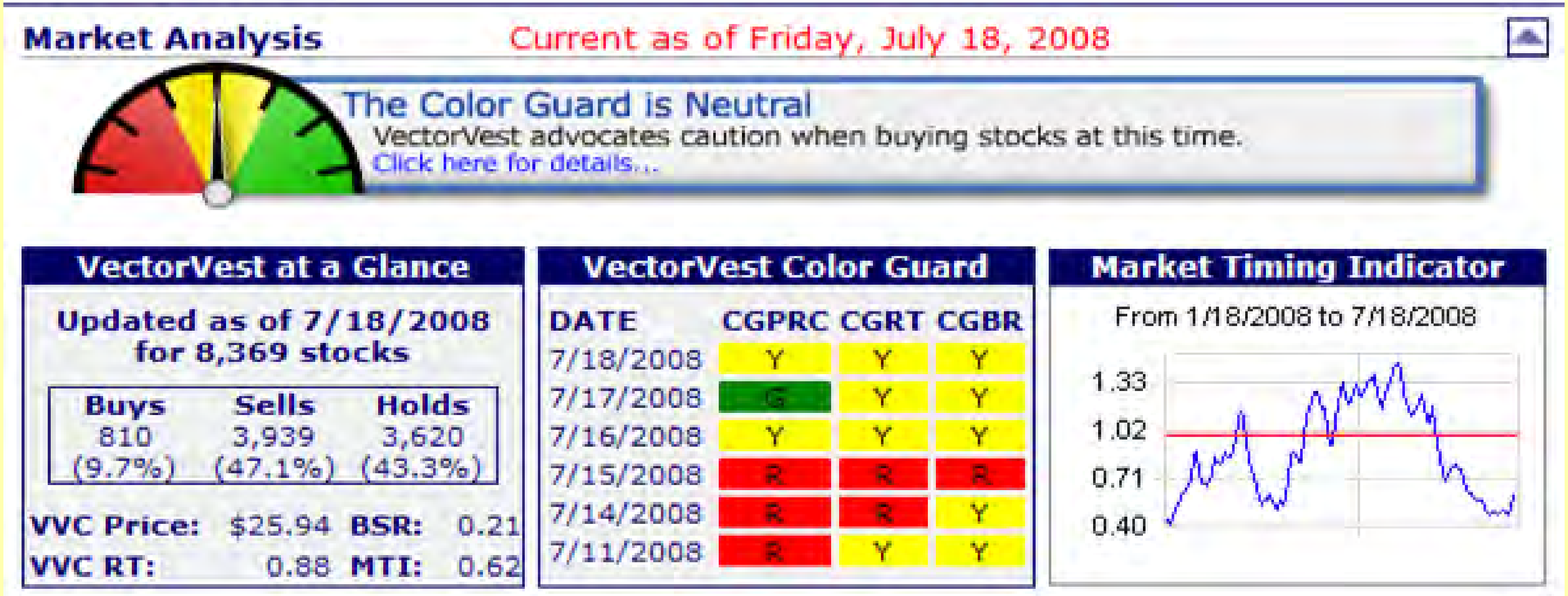
What if all 250 were no better than a Motley-Fool dartboard? How many should perform statistically significantly better than the market?

(Answer -- 12.5, at the p <= .05 level of significance)

Hmm...
What should
we call the
recommendations
that **didn't** make
money?

Maybe we should do some research

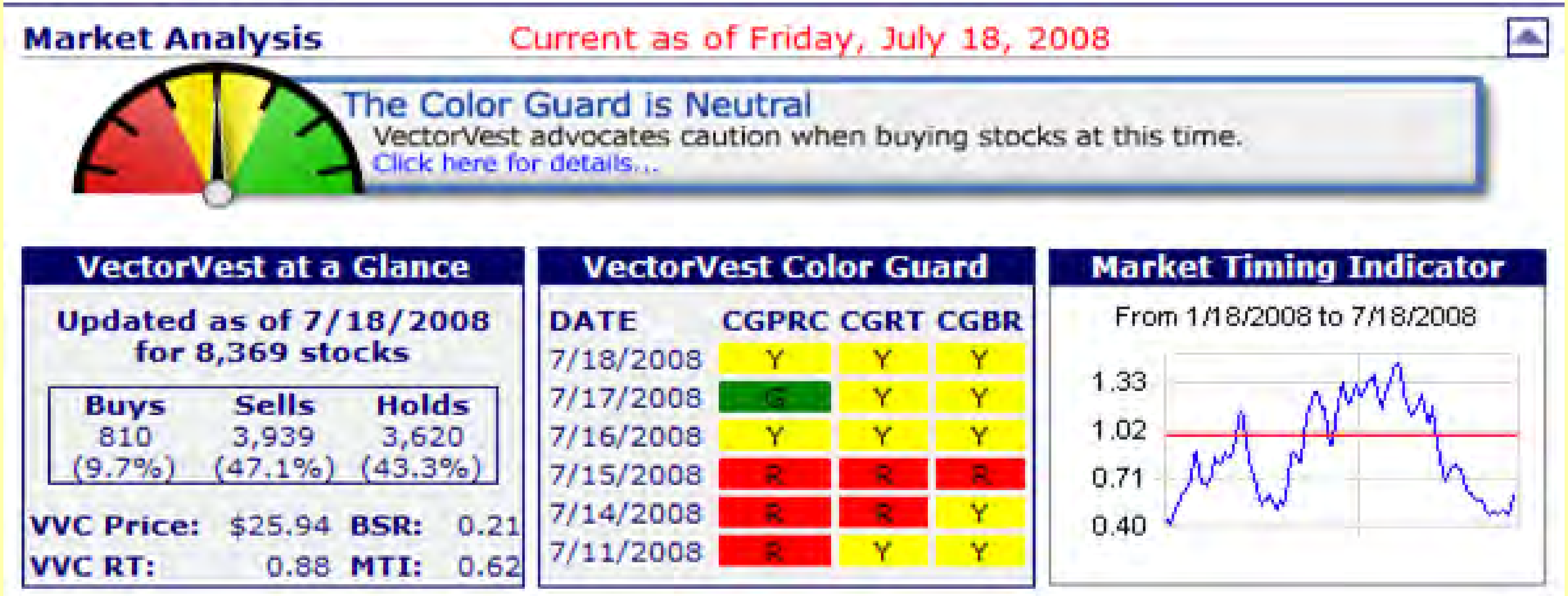




"COLOR GUARD" on the front page seems to be VectorVest's most unique and important feature. This box speaks to the overall timing of the market:

- VVC Price is the average price of the vector vest composite (the 8013 stocks in the VV database). They use it as an index, like S&P or Dow Jones
- VVC RT is the "relative timing" of the market. The market is on a rising trend for RT > 1 and a declining trend for RT < 1. Based on its published formula (ratios of random variables), I would expect this to have an odd probability distribution.

- BSR is the ratio of the number of stocks rated buy to the number of stocks rated sell in the VV database – ignoring the number rated hold. Suppose VV puts “hold” ratings on 8008 stocks, a Buy rating on 4 and a sell rating on 1, I would think this is a flat market, but with a 4-to-1 ratio of buys to sells (ignore the 8008 holds), BSR would have a value of 4.0, a seemingly huge value.
- MTI is the overall Market Timing Indicator and the is described in VectorVest tutorials as a key predictor in their system.



- The color system is summary of trends over the past few days / few weeks.
- According to VectorVest:
  - yellows mean there is no trend to follow,
  - red means the market is declining and you should NOT buy anything tomorrow, and
  - green means the market is rising and you feel good about buying.

VectorVest in their own words

Dr. DiLiddo on VectorVest - THE COLOR GUARD, CLARIFIED - Windows Internet Explorer

http://www.vectorvest.com/BLOG/post/2008/09/12/THE-COLOR-GUARD-CLARIFIED.aspx

vectorvest colorguard

Favorites | VectorVest - The Intelligent ... | Dr. DiLiddo on VectorVes...

THE COLOR GUARD, CLARIFIED

By Dr. Bart DiLiddo

Friday, 09/12/2008

I have received some feedback recently that the Color Guard is confusing and hard to understand, so this essay is dedicated to explaining, as well as I can, how it works.

Think of the Color Guard as you would a traffic light. Green means go, it's OK to buy stocks. Yellow means caution, it may or may not be OK to buy stocks. Red means stop, don't buy any stocks.

Can it really be that simple? In a perfect world, the answer would be yes, but there's more here than meets the eye. For example, I'm not going to go through a green traffic light if I see a truck in my path, running a red light. Even though I have the green, I always look both ways to make sure it's safe to proceed. This is why our trend indicators, the Primary Wave and the Market Timing Indicator, are important parts of the Color Guard.

We use colors to report the up and down movements of the Price of the VectorVest Composite, the Relative Timing of the V V C and the Buy to Sell Ratio, and we use Up and Dn signals to report the direction of our Trend Indicators. Put it all together and we have a set of three lights and two indicators. Here's a summary of the most likely combinations of light colors and trends, what they mean and the suggested action to take:

LIGHTS.....	TRENDS	MEANING...............	ACTION
3 Green.....	UpUp	The Color Guard is Bullish.........	Buy...
2 Green.....	UpUp	The Color Guard is Somewhat Bullish	Buy...
1 Green.....	UpUp	The Color Guard is Mildly Bullish..	Buy...
1 Green.....	UpDn	The Color Guard is Mildly Bullish..	Buy w Caution.

Search

Enter search term | Search

Include comments in search

Subscribe

Page List

RecentPosts

RETIREMENT STRATEGY - PART I
Comments: 0
Rating: 5 / 1

PRICELESS

VectorVest in their own words

3 Yellow....	UpUp	The Color Guard is Neutral.........	Buy w Caution.
3 Yellow....	UpDn	The Color Guard is Neutral.........	Buy w Caution.
3 Yellow....	DnUp	The Color Guard is Neutral.........	Do Not Buy.
3 Yellow....	DnDn	The Color Guard is Neutral.........	Do Not Buy.
1 Red.......	DnUp	The Color Guard is Mildly Bearish..	Do Not Buy.
1 Red.......	DnDn	The Color Guard is Mildly Bearish..	Do Not Buy.
2 Red.......	DnDn	The Color Guard is Somewhat Bearish	Do Not Buy.
3 Red.......	DnDn	The Color Guard is Bearish.........	Do Not Buy.

Given this array of light combinations, trends, meanings and actions, I can see why some people may be confused by the Color Guard. But here's how to make it real simple: IT'S ALWAYS OK TO BUY STOCKS WHEN THE PRIMARY WAVE IS UP.

The Primary Wave, shown on the left side of the Trends column, is discussed at some length in my essay of July 9, 2004. The Underlying Trend, shown on the right side of the Trend column, is discussed in my essay of July 23, 2004. A comprehensive review of our Market Timing System was presented in a series of seven essays from July 9, 2004 to August 20, 2004.

Comments: 0
Rating: 3 / 2

BUYING LOW AND SELLING HIGH
Comments: 0
Rating: 5 / 1

ON THE ROAD AGAIN
Comments: 0
Rating: 5 / 1

BECOME A GREEN LIGHT BUYER
Comments: 6
Rating: 3.4 / 19

VECTORVEST OPTION TOOLS
Comments: 0
Rating: 0 / 0

ONE DAY AT A TIME
Comments: 1
Rating: 3.7 / 3

THE YELLOW BRICK ROAD PORTFOLIO
Comments: 0
Rating: 5 / 1

MANAGING THE YELLOW BRICK ROAD PORTFOLIO

from ... THE COLOR GUARD, CLARIFIED

Bart DiLiddo (VectorVest founder)

A little research

To study the ColorGuard system as a predictor of the market, I downloaded Standard and Poors' S&P-500 index prices from January 4, 1999* through early Sept 2009.

I then computed percentage price changes:

- percent gain or loss in the S&P compared to the current day
- percent gain or loss between the current day value and the value 5 trading days from the current day.
- after 15 trading days
- after 30 trading days.

I also looked at 2-day, 3-day and 4-day for some analyses, but the results were the same as 1-day and 5-day so I stopped bothering.

- The average day-to-day change in the market was 0.0027% (flat over 10 years)

* Available ColorGuard data appeared to start in late 1998

Results

From 1999 to September 2009:

- 289 trading days rated **GREEN** / **GREEN** / **GREEN** (**GGG**) (**buy**)
- 285 rated **RED** / **RED** / **RED** (**don't buy**).
- After **GGG** days, S&P index went **DOWN** an average of
  - **-0.05%** the day after a **GGG** rating,
  - **-0.24%** five days after,
  - **-0.14%** 15 days after, and
  - **-0.33%** 30 days after.
- After **RRR** days, S&P went **UP** an average of
  - **0.24%** the day after an **RRR**,
  - **0.29%** five days after,
  - **0.42%** 15 days after and
  - **1.08%** in the 30 days after an **RRR**

Time to think about testing

Imagine these roles:

- individual investor trying to avoid making terrible mistakes and losing all her retirement money
- technical expert, hired by Securities & Exchange Commission to help them investigate stock-market related fraud
- analyst helping an investment firm assess effectiveness of investment strategies
- individual investor trying to assess effectiveness of investment-guidance tools ***that he is writing***
- analyst working at an investment firm that creates/sells investment-guidance tools:
  - writing the code
  - evaluating the usability and basic dependability of the product
  - evaluating the effectiveness of the product

These are all testing roles

Simplistic, but commonly recommended, buy/sell rules

- Buy in the early morning (market open), sell at market close (in a generally rising market)
  - (How do you know if you're in a generally rising market?)
- Buy at close, sell the next morning (in a generally declining market)
- Buy a stock when it drops more than D% in one day and sell it back when the stock regains ½ of its loss
- Buy a stock when it hits a 52-week high and sell (a) after B% (e.g. 25%) rise or 10% trailing stop

These are probably all losers, but how can we tell?

Assessing a Model

1. start with a plausible hypothesis
   - in practice, this is the hardest step and the one that requires the most investigation
2. decide what data to test it on
3. get the data
4. what's the right test?
5. if the model proves itself wrong
   - study the fine grain of the data for evolution to next model
6. if model appears right
   - what replications are needed, on what data, to check this further?

1. Start with a plausible hypothesis

- in practice, this is the hardest step and the one that requires the most investigation
- For now, let's select this heuristic:

In a generally bullish period, buy at market open, sell at market close

(reflects market optimism?)

2. Decide what data to test it on

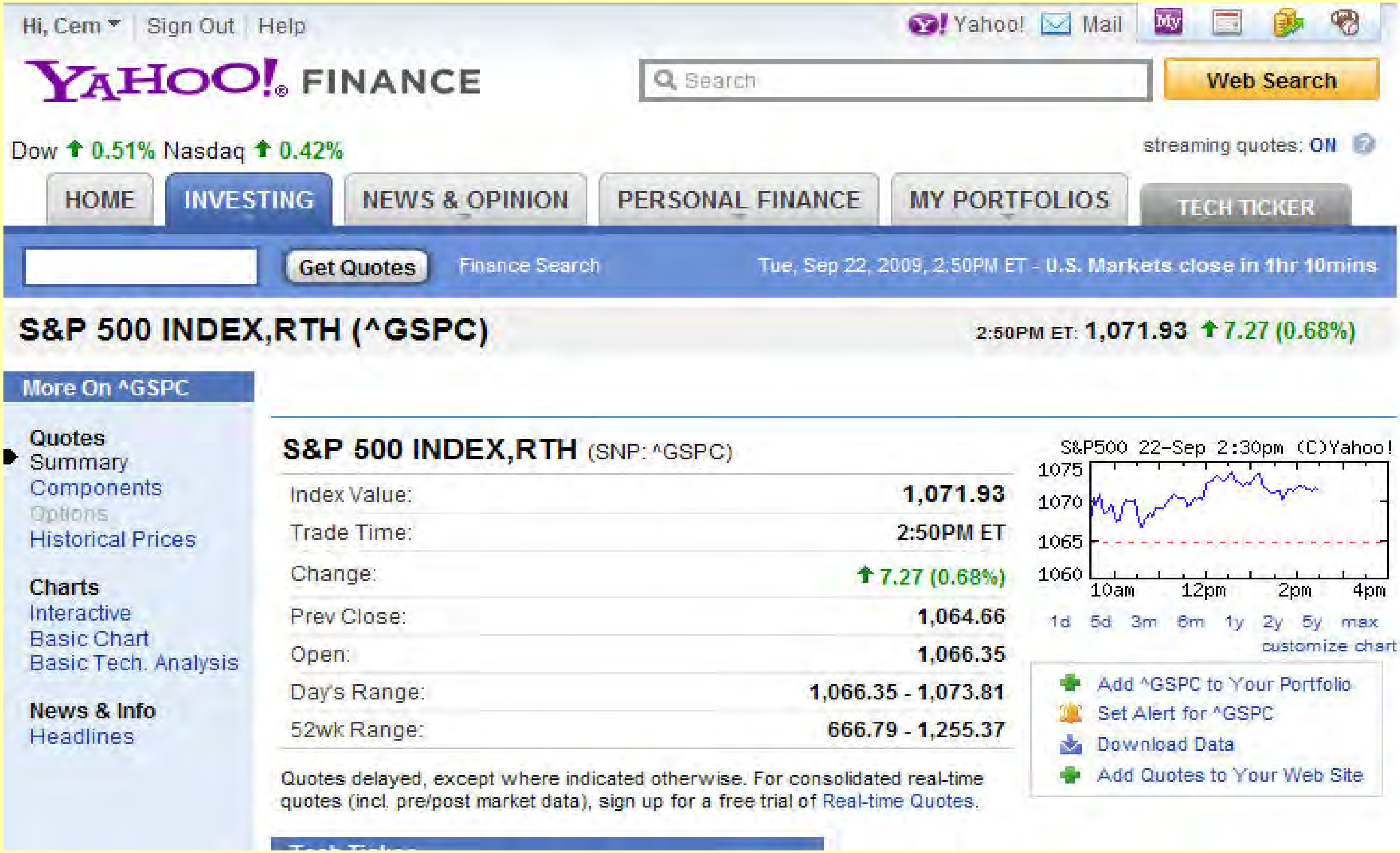
- Should we test on individual stocks or an aggregate?
  - If individual, should we sample from a specific pool (e.g. Wireless Internet stocks (think iPhone) might behave differently from consumer stocks like Taco Bell)?
  - How should we select from the pool?
- What time interval should we test on?
  - generally rising?
  - generally falling?
  - based on indicators (like consumer confidence)?
  - random?

2. Decide what data to test it on

- To simplify our example, let's choose the Standard & Poors index
- from March 9 to present (when I wrote this, September 2009)

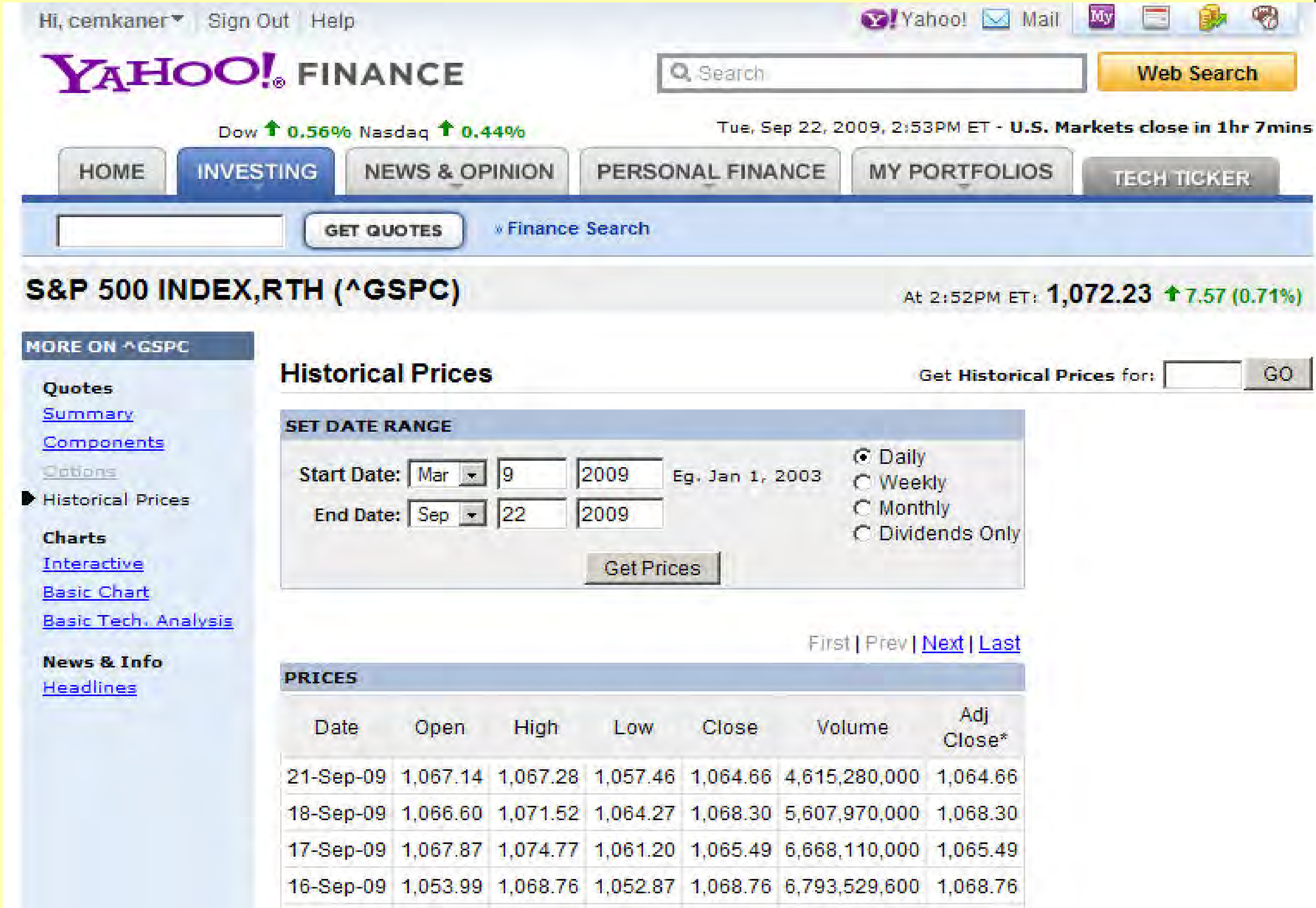
3. Get the data

- In this case, getting the data is easy.
- Yahoo has it.



3. Get the data

- But how do we know that it's right?
- (This is a serious problem across different data sources)



4. What's the right test?

- My test is to calculate

  delta = Closing index value - Opening index value

  for every day in the time period

  and then calculate average delta per day

Date	Open	High	Low	Close	Volume	Adj Close	CLOSE - OPEN	TOTAL	AVERAGE GAIN
9/21/2009	1067.14	1067.28	1057.46	1064.66	4.62E+09	1064.66	-2.48	407.27	2.97
9/18/2009	1066.6	1071.52	1064.27	1068.3	5.61E+09	1068.3	1.7		
9/17/2009	1067.87	1074.77	1061.2	1065.49	6.67E+09	1065.49	-2.38		
9/16/2009	1053.99	1068.76	1052.87	1068.76	6.79E+09	1068.76	14.77		
9/15/2009	1049.03	1056.04	1043.42	1052.63	6.19E+09	1052.63	3.6		
9/14/2009	1040.15	1049.74	1035	1049.34	4.98E+09	1049.34	9.19		
9/11/2009	1043.92	1048.18	1038.4	1042.73	4.92E+09	1042.73	-1.19		
9/10/2009	1032.99	1044.14	1028.04	1044.14	5.19E+09	1044.14	11.15		
9/9/2009	1025.36	1036.34	1023.97	1033.37	5.2E+09	1033.37	8.01		

table

Woo Hoo!!

5. If the model proves itself wrong

Can we strengthen the predictable-rise-during-the-day hypothesis?

- study the fine grain of the data
- decide whether the hypothesis is wrong or incomplete
- if incomplete:
  - vary conditions as (potentially) appropriate
    - If the underlying theory is a daily rise due to optimism
      - should we buy only when Consumer Confidence is up (we do have historical data)?
      - should we focus on stocks recently upgraded by analysts?
      - what else could enhance a general optimism, increasing its impact for a specific stock or industry or sector?
      - What if we tried EVERY variable?

6. If the model appears right

What replications are needed, on what data, to check this further?

- Replications on rising markets in previous years?
- Replications on falling markets?
- Replications on broader market (S&P is a 500-stock subset of 15,500 stock market)
- Replication across geographic segments (Chinese stocks? Israeli stocks? UK stocks?) (Do these add noise that should be chopped from our buying strategy?)

CAUTION: From my limited study, 2009 appears anomalous.

As far as I can tell, betting on this in the future is as likely to yield Boo Hoo as Woo Hoo.

Another Example: Buying the Dividend

Buying Dividends: Top 7 Stocks Going Ex-Dividend Next Week

One way that you can earn very high returns is utilizing a technique called 'buying dividends'. Buying dividends is the process of buying a stock just before it goes ex-dividend and selling it shortly after the ex-dividend date at the same [or at a higher] price as the cost price. The ex-dividend date is the date after which a buyer of the stock is no longer entitled to the dividend. In other words, if an investor purchases a stock on or after the ex-dividend date, the investor will not receive the dividend; however, if the stock is purchased prior to the ex-dividend date, the investor will receive the dividend on what is called the payment date. Another way to look at it is an investor can sell a stock on the ex-div date or later and still get the dividend.

Does it work? Let me give you an example. Last month, I bought United Online Inc. (UNTD) for 10.89 per share on April 11 and sold it on May 12, the ex-dividend date, for 11.56 per share, a return of over 6% before commission and before adding in the dividend, but that's really not relevant. The stock pays a dividend of 20 cents per share, which provided a return of 1.8% over 30 days. Annualized, it worked out to about 22%.

But I need not have tied up my money for a month. I could have bought the stock for 11.31 [closing price] on Friday, May 9, and sold it Monday, May 12, for 11.91 [closing price]. The gain on the stock was 60 cents, plus the 20 cent dividend is a return of 80 cents or 7% over a period of three days [before commission]. The annualized return is ridiculously high at 860%. But even if you broke even on the purchase and sale price, just the return on the dividend provided a 215% annualized return.

http://stockerblog.blogspot.com/2008/05/buying-dividends-top-7-stocks-going-ex.html lose?

Capital Product Partners:

Typical dividend: 41 cents. Typical decline: $1 to $2

What do we do with this?

- When I teach investment modeling, I have my students try to implement a buy-the-dividend strategy.
  - They lose “money”
- Next, they search the market for differentiators that can separate high-dividend stocks that are:
  - Buy the dividend candidates
  - Sell the dividend candidates
  - Stocks with no useful pattern
- As an easy differentiator to consider:
  - CPLP pays out more than it earns. Each quarter:
    - Before it announces the dividend, it’s hard to believe it will offer such a high dividend again (stock rises on announcement)
    - After it pays the dividend , it’s hard to believe it will offer such a high dividend again (stock falls on payment)

Why do we care?

- Each buy/sell rule reflects an underlying model of the market's historical behavior
- But in an "investment program", each implemented strategy is implemented as a feature of the program
- So if we are testing an investment program (financial services are now a huge part of our economy), we are testing each strategy

"Trailing stops saved me a pantload"

http://www.youtube.com/watch?v=mDm6iKH38C0

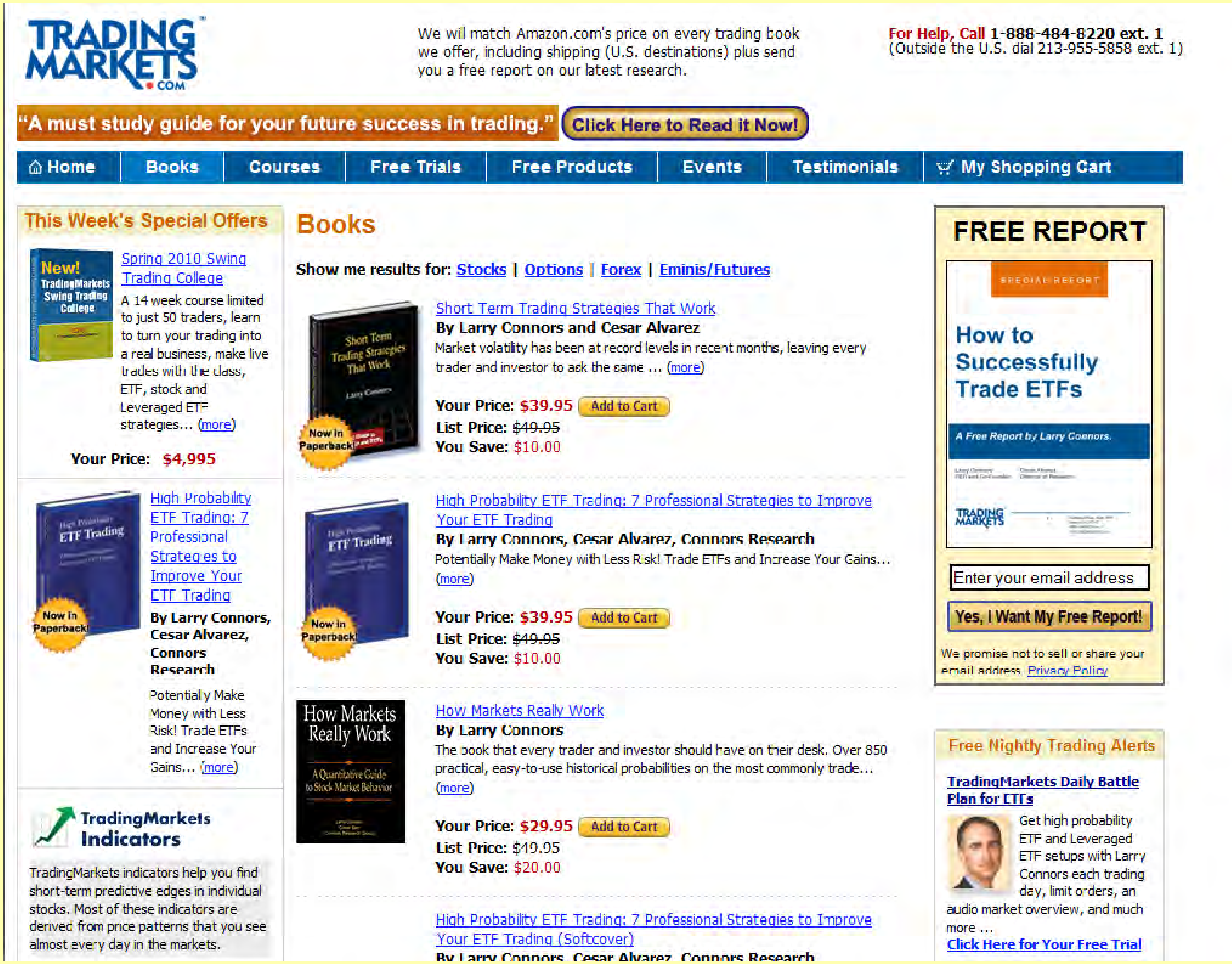
Trailing stops are generally set to sell at market price (whatever you can get at this time).

At any point in time, we have supply and demand for a stock.

- We can subscribe to services that show the backlog of orders for a stock. There might be orders for 1000 shares at $10.00, 1200 shares at $9.99, 300 at $9.97, 500 at $9.95, 200 at $9.90, 500 at $9.85, 1000 at $9.70, and 1000 at $9.00.
- If we offer 15000 shares "at market", we sell 5700 shares for $9.00 to $10.00 and are looking for buyers for the other 9300 shares. The sharp drop and the low price of the last trade ($9) invite more lowball orders, so the price keeps dropping until someone buys out all that's left.
- In the meantime, trailing stops get triggered as the price drops, adding to the supply of stock for sale and driving the price down even further.

Because of the prevalence of trailing stops, one big sell order across many stocks (S&P futures) might have triggered or contributed to a cascading automated selloff.

And how beneficial are trailing stops anyway?



Assessing Models: Seven Risks

Model risk	The model is theoretically incorrect.
Characterization risk	The model is correct, but the description of it (the specification) has errors.
Comprehension risk	We misunderstand the model. Our code accurately implements the wrong model
Implementation risk	Coding errors. Data storage / retrieval errors. Our code inaccurately reflects our intent.
Execution / environmental risk	We attempt the correct trade but the software / hardware platform is too slow, cannot handle the data volume, the data feed is too slow, etc. As a result, we fail in our efforts to buy or sell at the desired price.
Tool risk	Our test tool misleads us by corrupting the software under test or by missing failures or by giving us false alarms
Scope risk	Our model is properly developed but it is not appropriate to today's circumstances. Can we recognize when market activity is out of scope of our model?

Verification is insufficient

Model risk	The model is theoretically incorrect.	Validation
Characterization risk	The model is correct, but the description of it (the specification) has errors.	Validation of the software; Verification of the spec
Comprehension risk	We misunderstand the model. Our code accurately implements the wrong model	
Implementation risk	Coding errors. Data storage / retrieval errors. Our code inaccurately reflects our intent.	
Execution / environmental risk	We attempt the correct trade but platform is too slow, cannot handle data volume, data feed too slow, etc. As a result, we fail in our efforts to buy or sell at the desired price.	Parafunctional
Tool risk	Our test tool misleads us by corrupting the software under test or by missing failures or by giving us false alarms	
Scope risk	Can we recognize when market activity is out of scope of our model?	Validation

Let's stop the investment example here.

What does it show us?

The investment example

- When we study "computing" as a general field (or software testing) (or software engineering) we often abstract away the underlying complexities of the subject matter we are working in.
- A computer program is not just "a set of instructions for a computer." It is an attempt to help someone do something. The program:
  – makes new things possible, or
  – makes old things easier, or
  – helps us gain new insights, or
  – brings us new experiences (e.g. entertainment)
- Programs provide value to companies
  – Some programs tie directly to the core value-generating or risk-mitigating activities in the company
  – Especially in organizations that see computing as a technology rather than as a goal in itself, your value to the organization rises if your work actively supports the business value of the software.

Investment example

- Every evaluative step in this example involved testing
- This work requires extensive technical knowledge, but
  - "technical" = the conceptually difficult areas in the business domain
  - "technical" ≠ programming
- For years, the highest paid testers on Adobe Illustrator were graphic artists
- For several years at WordStar, the most senior tester was a printer expert
- Consider what testing you could do if you deeply understood:
  - actuarial mathematics (insurance risk modeling)
  - the technology (e.g. data mining) and psychology of customer relationship management and sales prospecting
  - taxation systems (e.g. as applied in employee compensation software)
  - airline reservation systems
  - the human factors of creative work (draw / video / text / music)

Now, Back to Automation

- Many people in our field are trapped in a mental box that says:
  - ***test automation Ξ automated execution of regression tests***
- This is a commodity role
  - automated execution of regression tests, is narrow in scope, redundant with prior work
  - The style of testing often promoted as "professional" is adversarial, inefficient, relatively unskilled, and easy to outsource
- In previous talks, I've emphasized "high volume test automation" as applied to such areas as telephony, printer firmware and other embedded control systems, and assessment of mathematical computations.
  - ***Lots of test automation, but no regression testing***
  - **http://www.kaner.com/pdfs/HVAT_STAR.pdf**

The Tasks of Test Automation

- **Theory of error**

  What kinds of errors do we hope to expose?
- **Input data**

  How will we select and generate input data and conditions?
- **Sequential dependence**

  Should tests be independent? If not, what info should persist or drive sequence from test N to N+1?
- **Execution**

  How well are test suites run, especially in case of individual test failures?
- **Output data**

  Observe which outputs, and what dimensions of them?
- **Comparison data**

  IF detection is via comparison to oracle data, where do we get the data?
- **Detection**

  What heuristics/rules tell us there **might** be a problem?
- **Evaluation**

  How **decide** whether X is a problem or not?
- **Troubleshooting support**

  Failure triggers what further data collection?
- **Notification**

  How/when is failure reported?
- **Retention**

  In general, what data do we keep?
- **Maintenance**

  How are tests / suites updated / replaced?
- **Relevant contexts**

  Under what circumstances is this approach relevant/desirable?

What level are you working at? (Some Examples)

CHECKING	• Testing for UI implementation weakness (e.g. boundary tests) • Straightforward nonconformance testing • Verification should be thought of as the handmaiden to validation
BASIC EXPLORATION	• Quicktests • Straightforward tours to determine the basics of the product, the platform, the market, the risks, etc. • Here, we are on the road to validation (but might not be there yet)
SYSTEMATIC VARIATION	• Conscious, efficiently-run sampling strategy for testing compatibility with big pool of devices / interoperable products / data-sharing partners, etc. • Conscious, efficiently-run strategy for assessing data quality, improving coverage (by intentionally-defined criteria)
BUSINESS VALUE	• Assess the extent to which the product provides the value for which it was designed, e.g. via exploratory scenario testing
EXPERT INVESTIGATION	• Expose root causes of hard to replicate problems • Model-building for challenging circumstances (e.g. skilled performance testing) • Vulnerabilities that require deep technical knowledge (some security testing) • Extent to which the product solves vital but hard-to-solve business problems

Closing Thoughts

- Especially in difficult economic times, it is important for:
  - testers to ask how they differentiate their own skills, knowledge, attitudes and techniques from commodity-level testers
  - test clients to ask how they can maximize the value of the testing they are paying for, by improving their focus on the problems most important to the enterprise
- In this talk, we looked at testing as an analytic activity that helps the other stakeholders understand the subject domain (here, investing), the models they are building in it, and the utility of those models and the code that expresses them.
  - **We see lots of test automation, but no regression testing.**
- Rather than letting yourself get stuck in an overstaffed, underpaid, low-skill area of our field, it makes more sense to ask how, in your application's particular domain, you can use tools to maximize value and minimize risk.
  - **High value test automation, probably not much regression testing.**