The introduction film was made three years ago, and it isn’t about the future, it is about now.
The industry believes testing is supposed to be about binary verification of stories, requirements, examples et.al.
I wrote this abstract for EuroSTAR 2011 and felt extremely pleased.
I had something new and important to say, but it wasn’t accepted.
So it became a lightning talk at SWET2, and a blog post, without an impact, as far as I know.
Finally I got to present it here in Malmö, maybe things will start happening now...

I might be very wrong, but I know I’m not.
You might be ill, and I might cure you.
My main message is that if you let go of these crippling computeresque ideas, you have time to focus on what is important.

You might think I’m attacking you, and you might think this has nothing to do with you.

Either way, think about your thinking, think about others thinking, and decide if there is a need for change or not.
It was a very strong feeling when I read Gigerenzer’s tools-to-theories (I found it via Gut Feelings). A lot of the testing theory have felt wrong, and know I could explain why.

This theory doesn’t automatically mean that all testing theories are bad and useless, but my intuition says it is more than a risk.
Have you ever seen a test case management systems without mandatory Pass/Fail?
Why do people do all of this pass/fail and metrics on it? Because that's the way you do it.

If we know everything in advance, then Pass/Fail might be OK.
But we never know everything in advance.

You can ask richer questions than is this correct or not? You can learn things, and grow as tester.
Have you ever seen a test case management systems without mandatory Pass/Fail?

PASS/FAIL REHAB

Do some deviations when executing tests
Look at some more places than what is stated in the Expected Results field
Write the occasional test idea using the word "investigate"
Put the numbers in smaller font in your status report
Observe the software without a hypothesis to falsify

You can ask richer questions than: Is this correct or not?
    You can learn things, and grow as tester.
See it as your daily medicine; eventually any Pass/Fail usage will seem ridiculous
COVERAGE OBSESSION

50% coverage can mean

* we have found so many serious bugs that further testing is pointless
* we are running late because testers insist on investigating things they aren’t explicitly told to look for
* we have run the 50 most difficult test ideas, and we believe we will finish on schedule
* we have run the 50 easy tests on input data, and look forward to the results from the radically different test ideas
* we have run the first half, in alphabetical order, and are not really sure what we are doing
* we have investigated the 50 most important test ideas, and believe the implicit coverage is enough to go Beta
* we are halfway through, but have found a lot of things that are more important to test than our original assumptions
COVERAGE OBSESSION

A coverage model is useful to get ideas
Not useful as a metric of completion

A model can help you find important things, but a percentage number might not include things that are important

Information about the system is more important than information about the model of the system (Emilsson)
At a few occasions I have had the chance to have “the talk” with testers I respect that advocate metrics.

It boils down to the same essence, you need A LOT of context for the numbers to be valid.

So much context, that I believe you could throw away the numbers and just keep the text.
I wrote a little book about test design that have received 5,000 hits in two months.
When someone challenges authorities, you should ask: “say you’re right, what can you do with this knowledge?”
The software potato:

The square symbolizes the features and bugs you will find with test cases stemming from requirements (that can’t and shouldn’t be complete)

The blue area is every possible usage, including things that maybe no customers would consider a problem.

The brown area is what is important, there lies those problems you’d want to find and fix.

This problem has been solved many times at many places with many different approaches. What is common could be that testers learn a lot of things from many different sources, combine things, look at many places, think critically and design tests (in advance or on-the-fly) that will cover the important areas.

Some part luck, and a large portion of hard work is needed. Serendipity is working to our advantage.

Recently I realized there can be more than one potato, that are three-dimensional and perhaps slippery; there might be small potatoes that are the best of them’all...
The potato is now filled with content from diverse sources.
This can be how the software world looks like for free testers.
We can investigate software as humans, make subjective judgments and handle the inevitable unknown.

We don’t have to look at things in binary ways, since that’s not what reality is.
Liberation

To set all testers free, you should start with yourself

First step is acknowledgement

Next steps are your own, but will include thinking in new ways

Might involve helping others trusting testers

Ask stakeholders: What do you really want to know?
  – three or four times if necessary.

Liberation of our thinking.

If you don’t trust testers, train them, so you can.
The purpose of testing isn’t to make it easy to create fancy reports, it is to communicate important information.

you need knowledge about details, and the ability to summarize and explain in a variety of manners.
you need to know the software details, because then you can say better things

If communicating the essence is too difficult, maybe there are other large-scale problems, and the project should be split in more manageable pieces.
COMMUNICATION

Do we know how to communicate the essence fast?
We must train analyzing and communication (for testing!)

We need more words, and better metaphors
  – saturation
  – quality has many faces
  – things connected to life, not machines
  – your appropriate words that build confidence and trust

A shared customized quality model can help
Discuss with stakeholders which characteristics that are most relevant in your situation. Let all testers have these in the back of their head.
GOING FORWARD

My steps are lighter since I cured myself

Testing isn’t easy
If you make it easy, you lose the best parts

Life isn’t about ticking off check boxes
It is much richer...
**QUESTIONS**

???

**References:**

- Adaptive Thinking (Gigerenzer)
- Software Testing is a Social Science (Kaner)
- The Little Black Book on Test Design (Edgren)

http://thetesteye.com/blog/  
Gigerenzer: Adaptive Thinking.
Kaner: Software Testing is a Social Science,  
http://www.kaner.com/pdfs/KanerSocialScienceTASSQ.pdf
Edgren: The Little Black Book on Test Design:  

rikard.edgren@qamcom.se  
QAMCOM Karlstad

*This work is licensed under the Creative Commons Attribution-No Derivative License*