13 naïve questions about test process deconstruction and other paranormal activities ...

by André Cloutier, M.Sc., PSM, SA
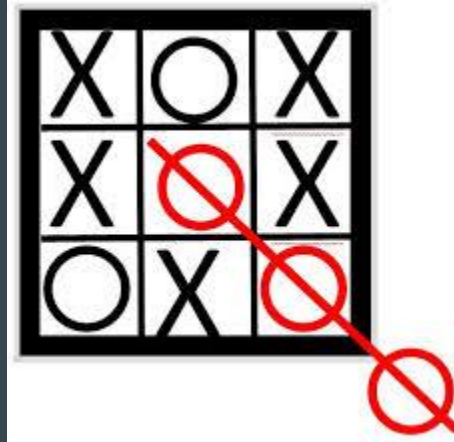
#1 Who am I?

#1 Ok... Who am I, really?

#2 Why am I doing this presentation?

Culture, Context and Agility

- Culture and Context has a huge impact on how agility is lived and understood
- Inertia (explicit or implicit processes rules, power games, etc.) is stronger than the Sunny Ways of the Transformation Forces...
- It's not different for test activities

What can we do?

What is acceptable?

What about "Agile" purity?

Deconstruction definitions

- *"A way of uncovering the questions behind the answers of a text or tradition"* (Paul Ricoeur)
- (more generally) Analytic examination of something to reveal its inadequacy

Deconstruction...

- ... exposes core complexities
- ... doesn't always lead to unconditional validation of best practices (aka what-they-say-in-the-book-or-blog-or-youtube)
- ... is Agile in essence (as Agile practices are themselves *countermeasures*)
- ... seeks the counter-intuitive
- ... shows that culture of the enterprise just cannot be ignored (as it may inhibit the value of orthodox practices)
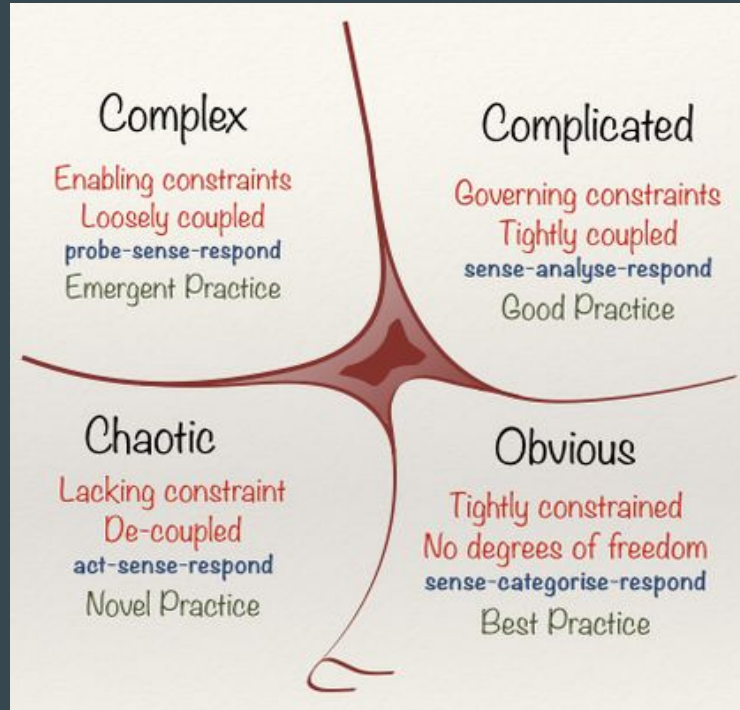
Software is *Complex* (allow enabling constraints) not *Complicated* (require governing constraints).

Dave Snowden - Cynefin

"If you don't understand why, you can't adapt"

Dave Snowden - Cynefin

Cynefin Framework

Complex
Enabling constraints
Loosely coupled
probe-sense-respond
Emergent Practice

Complicated
Governing constraints
Tightly coupled
sense-analyse-respond
Good Practice

Chaotic
Lacking constraint
De-coupled
act-sense-respond
Novel Practice

Obvious
Tightly constrained
No degrees of freedom
sense-categorise-respond
Best Practice

#3 Chris, why do you write all those Story Tests *upfront*?

Orthodoxy

- *If you are going to test something and document those tests, it costs no more to document the tests up front than it does to document them at the end. But these are more than just tests* (Pugh)
- Sutherland considers ATDD doubles productivity
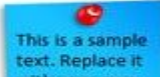
Some considerations

- ATDD
  - <u>Hypothesis</u>: Quality (through better specs) is higher, so waste is avoided
  - Assumes high collaboration
  - How much *time* could be saved otherwise though?
- Working Software
  - *"Often, you don't really know exactly what you want until you can see, feel, touch and use it."* (Crispin and Gregory)
  - [Manifesto] *Working Product over Comprehensive Documentation*
- Variability
  - Product can change during the sprint (e.g. analogy on a different scale with BDUF)
  - Low Cost of Delay: wait to the last possible minute
  - Waste in test authoring: tests with lower value need to be removed, some need to be reviewed
  - The Story is not a permanently valid artifact

Conclusions

- Orthodox
  - Consider Acceptance Tests as a way to increase communication with the Customer ("Triad")
  - Lean - Transparency
  - Mature culture without silos gets the full benefits
  - *Push* System for authoring
- Heretic
  - Focus *to the extreme* on Exploratory testing, become an expert (through *empirical* experience) on the feature, write only the optimal set of regression tests *after*
  - See ATDD as BDUF!
  - Revisit DoD (Story vs Feature)
  - Split test effort in different pieces, with more flexibility and optimized workflow
  - *Pull* System for authoring (or decoupled)

#4 Should we put time estimates to test tasks?

Orthodoxy

- Testing is just a "state" isn't it?

Some considerations

- For teams using time estimates for development tasks
  - Frequent complaints about dysfunctional accumulation of the functional test effort at the end of the sprint, especially if stories tend to be big
  - For burndown charts based on *dev* hours
    - Subtasks have time estimates
    - Subtask ownership issue
  - *Bottlenecks are frequent* (and hard to predict or avoid), so *blocking time* for testing may...
    - ...help sprint delivery
    - ...protect QA teams' weekends

Conclusions

- If testing is a *Task*, not just a *State*
    - Burndown Red Flags come earlier
    - Bottlenecks are easier to identify
    - Acting upon unrealistic plans can be done quicker
    - Basically *all* tasks (dev + qa) are estimated *independently*

#5 Can Automation of a Product itself be a Product?

Orthodoxy

No

#5 (retry) … or "Managed *like* a different product"?

Orthodoxy

No, you keep asking weird questions. I hate you.

Heterodox argumentation

- For Stories Sx and Sy
  - Effort for Sx: *Is Automation > Dev*?
  - Priority for Sx: *Is Automation != Dev*?
  - Quality Risk: *Sx > Sy*?
- "Product*ish*" then?
  - Requires planning and prioritization (project and product management)
  - Helps eliminating noise
  - What do you do if you can't keep the pace imposed by dev?

Conclusions

- Options
  - Synchronous (Push) - within Story / Sprint
  - Asynchronous (Pull) - within Sprint / Release
  - Put devs to work (Pull)
- Maintenance anyone?
  - When needs for fixing older tests collides with newer ones
  - Prioritization gets messy (and not even visible)
  - You can easily get overwhelmed
- Consider a separate backlog to fully own content and priorities
  - Is there an invisible bottleneck?
  - Scrumban (to alleviate variability in availability)

#6 Should Automation be on Stories or Features?

Orthodoxy

- For functional tests, automation is at the Story level. Timebox is the Sprint.
- Some non-functional tests may fit better at the Program level, so more likely outside the scope of the Story.

Some considerations

- Wait or Waste...
  - The feature keeps evolving across Sprints
  - Will those tests still be valid later?
- What is the Cost of Delay?
- What is the cost of simply not doing it?
  - It could happen if the priority is very low or *relatively* low
- Key point: flexibility in planning (not advocating laziness!)
  - "done" for the Release instead of the "Sprint"

Conclusions

- Story's scope is the sprint (e.g. 2 weeks)
- Feature's scope is the release (e.g. 3 months)
- Different DoDs!
- Flexibility and Pull

#7 "A Test Case is a Test Case"... What??

Orthodoxy

- Of course a Test Case is a Test Case! You write the test case, then you automate it, and then you check the box that says "automated". Easy. You eventually reach 100% and get a bonus if you do so.
- No... you should not automate everything. You know... some stuff is hard. But, you should automate *a lot* of your Test Cases though.

Some considerations

- I've heard way too many times that a Manual Test case and it's Automated counterpart are "equivalent", with all the weird consequences it can bring
- You automate your test and add the checkbox "automated"
- In practice, a lot of implicit checks are made in manual testing
- Problems that are naturally Data-driven. Cannot be "translated" directly from a manual TC
- The source of Automated TCs can not be the Manual TCs

Conclusions

- Two *independent* suites with different goals where duplication is tolerable
  - No need to be manichean... No need to *choose* between Good (Automation) and Evil (Manual)
  - Occasional grooming is good enough
- What if...
  - your automation works poorly and you don't have anything in your manual regression?
  - you want/need to outsource your manual regression?
- Can you guarantee that the automated tests are a valid replacement?

#8 What is your automation coverage (in %)?

Orthodoxy

- I think your M.Sc. in Mathematics was illegally obtained.
- So...
  - "A / (M+A) * 100.0"
  - Bingo...

Some considerations

- Different *Missions* require different *Workflows*
- How many fruits? Like counting and comparing apples and blackberries...
- Data-Driven tests may have lots of individual cases and create distortion
- Anti-pattern: "Management by objectives"

Conclusions

- It's a dumb metric
- It implies that both Manual and Automated TC are in the same pool, on the same scale, on the same planet...
- It's pervert...
  - It doesn't value quality of the selection of tests
  - It causes "automation blitz" to occur to get quickly to a higher percentage
  - It favors "quantity" over "quality"
- Replacements?
  - Metrics need to measure the <u>real value</u> of the work in automation
  - I really like the "bugs found with automation" metric (since you cannot hide your waste)
  - The need to measure progress can be harmful but yet required by non-agile management

#9 How can the resources efficiency be maximized?

(Agile Principle) *Simplicity—the art of maximizing the amount of work not done—is essential.*

Basically...

- Test the *implemented* stories
- Practice dissection on everything else
- Put all remaining work in queues!
- Keep it visible, where higher/lower priority is clearly distinguishable

Other considerations

Test Debt	Visible	Invisible
Workflow	Flexible	Imposed
Test Authoring	Regression Tests only	Story Test
Effort in	Spending / Investment - ROI	Fees
Priority	High	Low
Resource usage	Some slack	Full
Resource constraints	Generalists	Specialists

#10 DoD of Sprint or Release?

- QA fully owns the last hour of the sprint...
- Sprint boundaries are working well for development, but puts artificial pressure on QA. Scrum is cool, but the transaction cost of closing the sprint makes QA sub-optimal in both manual and automated testing activities.
- Fixing the boundary to the Sprint excludes certain flexible workflows
- DoD of Story or Feature? Similar question.
- Do before or do after?
- Synchronous or asynchronous?

#11 Does it smell like conclusion is coming ?

- Beware of anti-patterns (smells)!
  - Creativity is not limited, but *supervised* when you are aware of anti-patterns
  - Anti-pattern examples
    - Big Design Up Front (BDUF) - false sense of security
    - Not Invented Here - false sense of independence
    - One Size Fits All - false sense of control

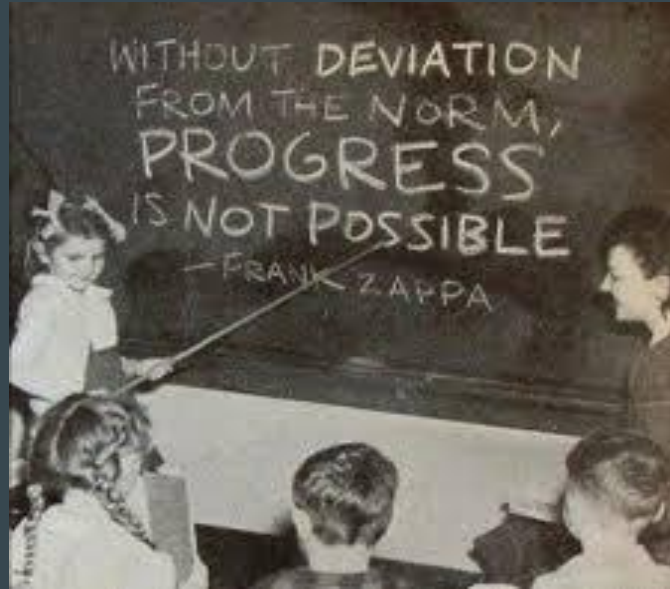Ask yourself if you are anti-patterning!

- Rules I try to follow
  - *Don't do* things differently just to do differently (avoid chaos of egos)
  - *Do* things differently just to do differently (avoid fatigue, you dig?)
  - If orthodoxy is costly, useless or sub-optimal: adapt rationally with coherence
  - Other smart people have usually thought about this problem before…

- Fundamentally...
  - Use independent queues for all aspects of the work where it's possible
  - Cut the work in independent pieces to see where the time is spent (where time can be saved!)
  - Unless you release at every sprint, consider the transaction cost of closing all your testing tasks at every sprint, targeting the release may significantly improve the productivity

Split...

Testing	and	Test Authoring
Manual TC regression	and	Automated TC regression
Product Backlog	and	Automation Backlog
Dev time estimates	and	Test time estimates
Work to be done in the *Sprint* (Working Software Testing)	and	Work to be done for the *Release* (all other activities)

#12 My motto?

#13 Questions?

Thank you!