Modern Testing (MT)

- principles
  - THINKING LIST - Place for notes without home
    - Leaders ship is disappointing people at level they can absorb
    - in martial art black belt is beging up on that point you can between what you need to learn
    - Modern testing principles not modern testers principles
    - Specialists is bottleneck
    - automation especially UI black hole of payoff
  - https://www.ergosquared.com/diff/testing/modern-testing-principles/
  - Why Do We have MT Principles?
    - WHAT EXIT ON WERE WORK THAT AS TWERE
      - MT is looking how to get THINK
        - Missing two from cost center to cost benefit
        - Being Force multiplier
          - LV World is changing and Traditional Testing is not catching up
            - Any time there is a change there is opportunity
    - Software is much easier to produce
      - Thanks to advancements in Continuous Delivery, the customer has much more options to choose from
        - Companies are shipping faster
      - It's much easier to compete with any company
        - For customers, the switching cost has gone way down
  - The principles are not just theory
    - They are based on Experience and observation
  - Why modern testing
    - To make contrast with Traditional
      - Traditional testing is not only a cost but also delay
  - Principle 1 — Our priority is improving the business.
    - What it means to improve the business
      - MTer needs to for seers as product influence
      - Building Frogs
        - Adoption has to be gradual
      - Dallas from idea to deployments is rapid
      - Can that step out with just enough quality to get feedback from customers
      - "You don't get value from engineers effort until it is in customer hands"
        - Think not only how to get it out but what's next
      - No one wins your product better than your customer
        - what in case of software Freeware? They don't have production or
      - There are bugs and bugs that customer cares about
      - The Aim
        - Short Term
          - Provable failure to customer Frequently
        - Long Term
          - Making sure Principle are Followed
    - Our goal is to accelerate team
      - that doesn't mean "we have to go fast"
        - Instead we are force multiplier
    - WHAT
      - Its ahead
        - adaptibility
        - Baseline
      - ??
        - Event seems on agenta and never finished his thought.
      - 1. Identify Value
      - 2. Map the Value Stream
      - 3. Create Flow
      - 4. Establish Pull
      - 5. Seek Perfection
      - Lean Principles
        - Deliver fast by managing the flow
        - Source: https://www.lean.org/WhatLeanAre/Principles.aspx
    - Books to read
      - The Goal: A Business Graphic Novel
      - Phoenix Project
      - Lean Thinking
    - WHY
      - Don't forge MT context
        - By focusing the MT Practitioner on identifying their role in an alternative?
          - this model name chosen because they are proven
      - It's important cause So to much of The TEST is reverse of this
        - they create the bottleneck instead of resolving them
          - Example stabilization period just before release
          - Its easier to "build quality in" instead of trying to test "quality in"
            - Test Complian dem
              - No i don't mean the test
              - "If you test after two weeks you will find bugs and then will need to rework the development"
    - HOW
      - This mentre is About completion and predictability and not about the Quality of software. It is a mechanism to identify what value is and what the next bottleneck
        - We don't want testing to be viewed as bottle neck
      - Quality is problem solved in humans being
      - Fast feedback loop is critical in context of understanding what is quality
      - By Balancing Velocity of value with failure in launch
  - Principle 2 — We are a force for continuous improvements, helping the teams adapt and optimize in order to quickly, safer than providing a safety net in production.
    - WHAT
      - "We are going to faster continuously adapting and improving where safety net presentation model"
        - They chall Drill your safety net
          - Traditional testing is treated as safty net
      - It's wrong that testers feel validated by the fact the developers don't testers mm
        - If we are going to more quality upstream, we have to remain relevant/remember the safety net
          - Too many people think the quality is limited to testing
      - The tester can coach code correctness, but does own
      - Code correctness
        - Developers have to learn from their mistakes.
          - Continuous improvement is about not being afraid to make mistakes, but embracing them as a learning opportunity.
        - Isn't unit testing a safety net too?
          - Yes is . And it should be there. TESTBEL aware through
      - Event harness based in developers stopped using mocks/stubs and mode doing unit testing. Cause automation would catch mistakes. - It's important to apply effectiveness can't scale and is hard to maintain.
      - Safty net is a problem
        - Example
          - To retrieve safety net, you need also very minimal risk of putting focus on functionalities or manage current be secure
      - Continuous Improvement
        - getting better all the time
          - Every time, "one" every half year, "two" every two weeks
            - ALL THE TIME
        - Antimatrix is sarcy.
        - Never
          - looking for improvement every time everywhere
          - reducing the waste
          - Efficiency vs productivity
    - WHY
      - Its a critical process change directly related to Quality
      - Testers are well suited to drive continuous improvement
        - Tester has to see the system as a whole because of this they can easily spot place for improvement
        - Testers are specialists in asking questions
    - HOW
      - By measuring
        - Visibility
      - unified engineering
      - cuts of small changes
      - Frequent retrospective (only 1-2 action)
        - Also useful - ended for long term - serious
      - A small vertical slices of architecture
        - MVP.
      - Master branch owned by the business. Bugs can release in advance. Hey want
      - Quality culture and leadership
        - Testers need to - stop being passive.
  - Principle 3 — We are always about the quality culture of our team, and we coach, lead, and nurture the team towards a more mature quality culture.
    - WHAT
      - Quality
        - is not bugs
        - is not code correctness
        - is user reaction to what you built
          - What is a higher-quality app with no bugs but with slow users, or app with lots of bugs but 100k users?
        - is about eliminating friction (enchanting pentina, enthusiasm, inspiration)
      - A
        - Case Profiteer
      - The shared mindset that delivering high quality software to the customer is our top priority and all our promises support this effort
      - B
        - Bad Profiteer
      - IT is how do we build healthy, actives, not intuition based, useful view of users empathy within the organization solving actual problems that users have
      - Quality culture
        - Getting people closer to understanding the problems that users are facing. How software developers are geared to solve it and how to stay curious
          - Getting people closer to understanding the problems that users are facing. How software developers are geared to solve it and how to stay curious
            - They, customer are you happier ?
          - It is mentally model use with coaster
            - Its alert model . It doesn't has to work for you
          - Testing breadth
            - infancy example
              - functional testing
            - adulthood example
              - contextual testing
          - Quality test ownership ship
            - infancy example
              - Testers are coaches
            - adulthood example
              - Tc coaters coached , occur team gets 4
          - What is mature Quality culture
            - Bug life cycle
              - infancy example
                - bug backlog
              - adulthood example
                - zero bug policy
            - how do you so code correctness tools
            - Test analyst
              - infancy example
                - oldstress
              - adulthood example
                - data centric
            - Development approach
              - infancy example
                - ad hoc
              - adulthood example
                - super small pushes
            - Learn and improvement
              - infancy example
                - no learning Happens.
                - retrospectives are for winning
              - adulthood example
                - every failure is opportunity to learn
                - and opportunity is taken
            - Leadership support
              - infancy example
              - adulthood example
                - good quality practices are treated supported and celebrated
              - read book
                - Book leaders life on the line
    - A — AWARENESS OF THE NEED FOR CHANGE
    - D — DESIRE TO SUPPORT THE CHANGE
    - K — KNOWLEDGE OF HOW TO CHANGE
    - A — ABILITY TO IMPROVE SKILLS & BEHAVIORS
    - R — REINFORCEMENT TO MAKE THE CHANGE STICK
      - source
        - https://www.prosci.com/adkar
      - ADKAR Model
        - Awareness
        - Desire
        - Knowledge
        - Ability
        - Reinforcement
    - WHY
      - The reason is our team feedback from our customers more clear
        - we want to understand and react to what they are strong,
          - we want to reduce risk in the ability to do that
      - To want is the way we want to function, to keep, retain and engage more users we need the quality culture
      - no mean testing
        - From Risk minimalization
          - in reaching to knowledge garnerd
      - Unless the whole team has a common understanding of the problem trying to solve it is pointless
    - HOW
      - Small Batches.
      - Feedback loop
      - Knowledge sharing (software is a knowledge works)
      - Try stuff + fail + learn
        - Experiment
        - Don't make sweeping changes at once
        - Small changes, and dont give up
          - Don't try to change all at once
      - Keeping the hand-crit works only when people are ready.
        - eg. Vistaray: mind time ab QA, then ou dac, now DS. - I don't know, so analogue. Half of my dev were still school we don't test and half were ex. Testers. - It's a long way to go, but we are on the right path. - Analogue to TDD, and long-lasting advocacy, as dead end. - Not nearly, long-lasting, bugs will not be fixed by the code author.
          - After learning/upskilling period was over, the whole team mentalmod they would not ever go back to old ways...
      - Brain example on slack TLDR series
  - Principle 4 — We believe that the customer is the only one who can judge the value of our work, and we work hard to ensure our product is loved and delivers high quality.
    - WHAT
      - Definition customer - it is a human who takes a decision that impacts our product. - It is not human can make decisions about quality.
      - It is not about pioneering teams
        - Free do not help business to go faster
      - Customers don't want software; they want the problem solved
      - It is radicalize to believe "hey I am doing it forever I know what I am doing
        - which is not often true
      - There is a risk in trust that QA battle in requirements dec and PMs, are solving a customer problem
        - Product owner developed areas we think they are cool, not because it solves problems.
        - Quick often asking teams.
          - Q: "What problem is solves?"
            - Iterative answer
              - "They will use it"
              - Q2 "When value does it bring?"
              - "They will use it"
      - Usually, you need to decompose the team to find the need
      - Need to want
        - Kids
      - Example of building wrong thing
        - Microsoft Kin
        - https://en.wikipedia.org/wiki/Microsoft_Kin
    - WHY
      - Ask your self
        - Who is my customer ?
          - What is the value when I am dedicating software to business?
            - Who is. customer that (Nicenessy or its a customer?
              - Kids
              - "You need to be able to help your customers help their customers.
          - Best users who benefit from software's you are producing
            - Testers are not customers
              - Unless you are selling software for testers.
      - Requirements from the business should be treated as hypothesis
      - For each feature request you can ask question
        - "Imagine you had that what would you do with it?"
    - HOW
  - Principle 5 — We are deep connected to deeply understand our customers, and we use this knowledge to design products hyperfocused and business aligned.
    - WHAT
    - WHY
    - HOW
  - Principle 6 — We expand testing abilities and knowhow across the entire development team, since quality is everybody's job, and we encourage the need for a dedicated testing specialist.
    - WHAT
    - WHY
    - HOW
- Alk Testing episodes to listen
  - Principle 1
    - S1 - E1
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-01-business-risk/
  - Principle 2
    - S1 - E3
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-03-unlocking-the-tester/
  - Principle 3
    - S1 - E2
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-02-unit-test-what-changes-and-how/
  - Principle 4
    - S1 - E5
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-05-the-community-principle/
  - Principle 5
    - S1 - E6
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-06-the-customers-champion/
  - S2
    - more principle 1
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-02-incidents-bugs-and-triangles/
  - Principle 6
    - S1 - E7
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-07-the-tester-as-a-leader/
  - principle 7
    - S1 - E8
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-08-testing-not-dead-testers-alive/
  - principle 8
    - S1 - E9
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-09-testing-in-an-modern-testing-principles/
  - general rule: is customer of the principles?
    - S1 - T7
    - S. Testers
      - https://www.ergosquared.com/diff/testing/alk-testing-episode-07-testing-in-an-modern-testing-principles/
  - S1 - T8
    - https://www.ergosquared.com/diff/testing/alk-testing-episode-08-testing-in-an-modern-testing-principles/
  - https://www.ergosquared.com/diff/testing/alk-testing-episode-05-modern-testing-means-context-driven-testing/
    - S4
      - Modern testing vs Context-driven
  - https://www.ergosquared.com/diff/testing/alk-testing-episode-04-modern-testing/
  - https://www.ergosquared.com/diff/testing/alk-testing-episode-03-test-quality-culture-transition-guide/
    - Transition guide
  - next
- FAQ
  - A: Agile is more fundamental than MT and addresses generic issues. MT principles are more focused on addressing testing issues
    - Q: "These principles sound like Agile things. Why are they called testing and not agile principles ?"
  - Q: Which of them sound more like managers' principles than testers
    - Are testers
  - Reducing waste and increasing value
    - There is no MT specialist. But MT activities are part of unified development. MT practitioner see value PM and testers see a lot of waste
  - Influence of MT mindset principles
    - unified development definition
  - https://leanpub.com/testers-mm/2013/01/23/the-unifiedmind-of-engineering-software-crowd/
- How it applies to polish market?
  - See principle #3 - B2B
- Historical information
  - Episode 77 - P
    - General
      - Based on agile principles - P
        - 1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
        - 2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
        - 3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
        - 4. Business people and developers must work together daily throughout the project
        - 5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
        - 6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
        - 7. Working software is the primary measure of progress.
        - 8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
        - 9. Continuous attention to technical excellence and good design enhances agility.
        - 10. Simplicity—the art of maximizing the amount of work not done—is essential.
        - 11. The best architectures, requirements, and designs emerge from self-organizing teams.
        - 12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
      - Who is customer for MT Principles?
        - Forward in this version
          - Tester as coach
        - Modern Tester
          - He is not Champion of quality
          - Responsible for quality culture
        - Are think of are equally important
        - Principle representer
      - Does Unless principle 1, is most important
      - Principlesa line version
        - 1. The customer is the only one available to judge and evaluate the quality of our software.
        - 2. We apply the theory of constraints to identify, exploit, and elevate the weakest element (not the system) and unstoppable the team.
        - 3. We are not the safety net for software correctness. I let focus on improving the business, and not the code.
        - 4. We are responsible for the quality culture of our team, and we coach, lead, and nurture for healing and coaching the team in this regard.
        - 5. We use data to deeply understand customer usage and customer pain.
        - 6. Embrace continuous improvement, and help the team adapt and optimize in order to solve customer pain.
        - 7. We strive to reduce or eliminate the need for a dedicated testing specialist on our teams by increasing testing abilities, and knowhow across the team.
    - Legacy current version doesn't have modern tester role per se
      - P
    - Legacy Principles
      - Software quality costs overrunner
        - It's a combination of Testing Discipline , agile and Business as data
        - Testing is not responsible for code correctness
        - Testing often is a double-next Bar only in a non-cause or one.
        - Consumer to evaluate of beetle necks
      - As about ballerina
        - neither it doesn't need to be now, nor it doesn't have to be perfect. balerina
  - https://www.ergosquared.com/diff/testing/alk-testing-episode-01/
    - traditional vs modern debate
    - Legend
      - Answer that is more funny / re...
      - Outdated
      - Quote
      - Needs more inducing questions
      - Attention
      - Materials already added to map