# Should / Could all testing be Agile?

## Keynote at Turku Agile Day, 2009-03-19

**Erkki Pöyhönen**
Business Development Manager
Tieto Corporation
Global Testing Practice
erkki.poyhonen@tieto.com

# Should we repeat these?
# Agile manifesto: Agile values

- Actual values are enticingly simple
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- They don't say anything about *how* you are supposed to really do the testing in agile paradigm
- …
- Oh, it does not even say how software in general should be developed in agile paradigm (actual practices)
- This is the beauty of the whole agile manifesto: it says any operation that agrees with agile values and principles is agile
  - Practices are secondary to agile values and principles

# Some background

- Erkki Pöyhönen:
  - Developer (programmer, tester), facilitator, trainer, consultant
  - Started development career in the happy 80's as programmer in a small plan-driven company, I was expected to be the most flexible part of the process (to ridiculous levels) – later learned that there are limits you can make even programmers work per day
  - Mostly telecom R&D background, with exposure to internal IT projects at construction, telecom and insurance domains
  - Once thought that all testers should be engineers (sad, huh?)
  - 1st real agile exposure in internal tool XP project around 2001
  - At the moment supporting projects – and supporting testers serving projects on several customer contexts
  - Believes that "Capital-A" dogmatism is not good for the field
  - Reading now the Gerry Weinberg's "Perfect Software" and "Agile Testing" by Lisa Crispin and Janet Gregory

# Agile manifesto: Agile principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in develop… …esses harness change for the customer's competitive ad…
- Deliver working software frequently, fro… … of months, with a preference to the …
- Business people and deve… …roject.
- Build projects ar… …nd support th…
- The … … within a dev…
- W…
- Agi… …he sponsors, developers, and user… …pace indefinitely.
- Cont… …xcellence and good design enhances agility.
- Simpl… …izing the amount of work not done – is essential.
- The bes… …es, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

**But how about testing in plan-driven organisation?**

# Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- Plan-Driven view:
  - "Early" normally means unstable, something that will potentially change
  - "Continuous delivery" means lots of regression testing
- "Valuable software" sounds like relevant quality has been reached
  - Definition of "Done" is critical at any context
  - "Don't trust the development" is not a realistic option
- Also early delivery of valuable software means there is explicit prioritisation in place
  - Needed also in plan-driven paradigm, especially with large integration risks (new platforms, new team, …)

# Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Plan-Driven view:
  - Late and too many changes is the second most common complaint of testing people in IT context (after development schedule drift)
    - Lessons learned of 28 R&D projects
  - In plan-driven paradigm the change is by default always bad
    - Some test plans and test design (test cases, test data, environments, ...) are built for certain assumptions in mind
    - If these assumptions change later some rework will be needed
    - How much can we risk wasting effort making same thing on several rounds
  - Of course, plan-driven is not same as waterfall
    - But good scope management in iterations is not very common
  - Most of “the change” is actually “the learning effect”
- Testing should be the one who recognises customer value, even if it lowers efficiency
  - Agreeing that all is not of same importance requires trust !

# Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

- Plan-Driven view:
  - Integration capacity can become a bottleneck
  - Technical change management is a common problem in incremental delivery: SW build and so also testing environment breaks as some unmanaged changes are not necessarily well tested in development context
    - Testing depends heavily on mature configuration management
    - Having covering automated tests for smoke test after integration would be a great benefit; systematic TDD is rare in plan-driven world
- (see "continuous delivery" above)
- Trust again
  - Stephen M. Covey: “Speed of trust”, son of Stephen R. Covey of the “7 Habits” fame

# Business people and developers must work together daily throughout the project.

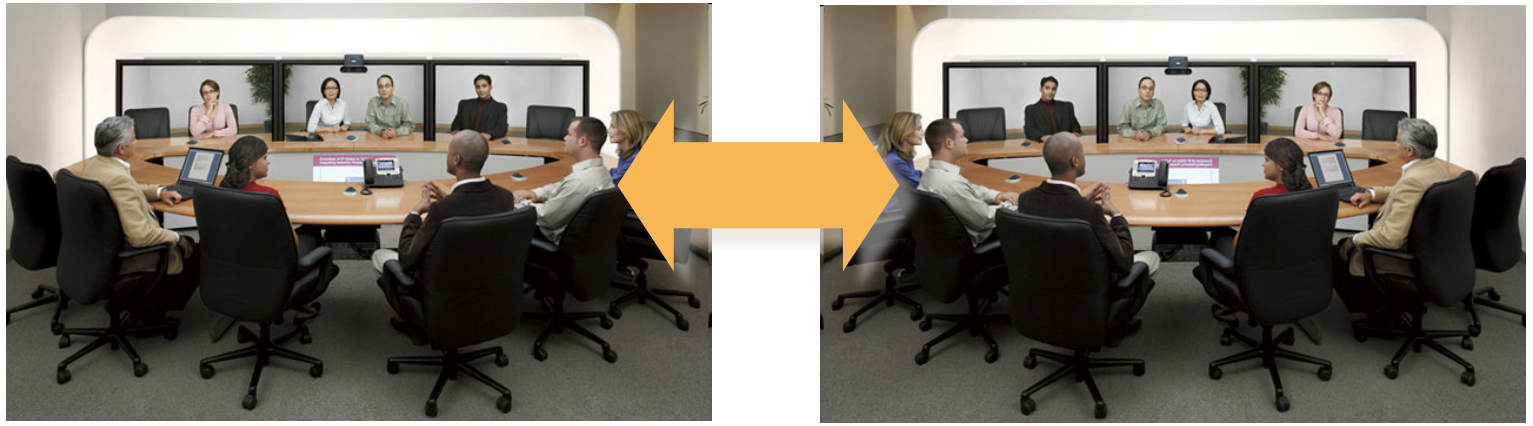
- Plan-Driven view:
  - In theory test strategy is built on known priorities at start but it is common that they change during the project
  - The best examples had common strategy calibration between concepting (requirements + process), development and testing, to make sure all had priorities straight
  - In product development the co-operation is frequent in last quarter of a project, but unfortunately seldom better than:
    - Marketing guy: *Are we there yet?*
    - Tester: *Nope, not yet.*
    - Think about kids in the car back seat during a long-haul drive? ☺
- "7 Habits of Highly Effective People" by Stephen R. Covey:
  - Habits 4-6: "Seek first to understand and then to be understood", "Aim Win/Win" & "Synergize" – Very agile!

# Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- Plan-Driven view:
  - All projects would benefit from having senior and self-driven individuals
  - Common thinking in process circles: we have to make it possible for “Average-Jack-O-Programmer” to provide consistent results
    - … via oversimplification, bureaucracy, micro-management, …
  - Trust is critical in knowledge intensive work – command & control does not work in general SW projects
  - "Trust the dealer but still cut the cards": Visibility is required nevertheless for real trust
- At any paradigm: Whose message is being valued?
  - Testing must always earn the trust and respect by providing relevant information, not because of position as gatekeeper

# The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Plan-Driven view:
  - In large and complex projects distribution (geographical and organisational) is more of a norm
  - Video-conferencing can really do wonders but is still not the same
    - A continuous video-conference between Helsinki and Copenhagen provided great support for an intensive distributed project
    - Not required by the process nor practices but by the people
    - About trust: SW organisation is a meritocracy, not position-based



# Working software is the primary measure of progress.

- Plan-Driven view:
  - Major conflict between test manager and project manager is how ready we are
  - Especially risky if management assumption of testing is analogy of "rubber-stamp the OK onto the system"
- Especially relevant if whole team shares the meaning of "working“, or can talk it over when differences appear
  - Systematic technology-facing tests (Brian Marick) are key for making this principle to work
  - Main measure is how team and individuals react to bugs
  - Luckily agreeing on business-facing tests is not normally not constrained by team creativity

# Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- Plan-Driven view:
  - Seldom achieved
  - Testing is even assumed to work extra when developers work 9-to-5
  - Few times phone rings on Thursday evening and a customer voice asks: *“We’re going live on Monday morning with new release, could you get it tested before that?”*
- The "Boy scouts" camping principle applies: "Leave the camping grounds and facilities in at least as good condition it was when you got it for yourself"
  - Can we apply this for our people and environments in all contexts?
  - Is the company culture preventing us?
  - Low organisational status of testing as a function or team makes things worse
  - A way to make average developers to senior developers, without loosing them in the mean time

# Continuous attention to technical excellence and good design enhances agility.

- Plan-Driven view:
  - Change management stands on the solidity of product architecture and managed dependencies
  - The value and feasibility of two-way traceability, and to what detail?
  - Technical excellence can imply wide range of documentation thoroughness (depends on use and users of testware)
    - Some testware can act on double-duty as metrics or models
- Also testware should be shared and refactored
  - If some testware assets becomes “owned” by someone you create problems
  - Not all testware should be built to save and archive, tying down resources

# Simplicity - the art of maximizing the amount of work not done - is essential.

- Plan-Driven view:
  - Prioritisation in scope, tight scrutiny in chosen practices and waste in bureaucracy
  - Good information flow avoids loosing important process and product information when integrating testing-to-development into a common product process
    - Avoiding silos is harder in distributed work (development & testing on different sites or even companies)
  - A good test strategy / approach is instrumental in avoiding waste in any method / paradigm

# The best architectures, requirements, and designs emerge from self-organizing teams.

- Plan-Driven view:
  - Applies also to testing: we commonly have to over-test first in order to find testing types and sections where we can optimise
  - Also gaps of our testing and development strategy are found from systematic faults in production
  - Like in typical process growth models they go from effective to efficient, not the other way around
  - How do you allow “self-organising” in partly subcontracted development?
- Most organisations seem to go through several stages of centralising, specialising, generalising and distributing until they find a working solution – for a while
  - Frankly you need both distribution and centralisation

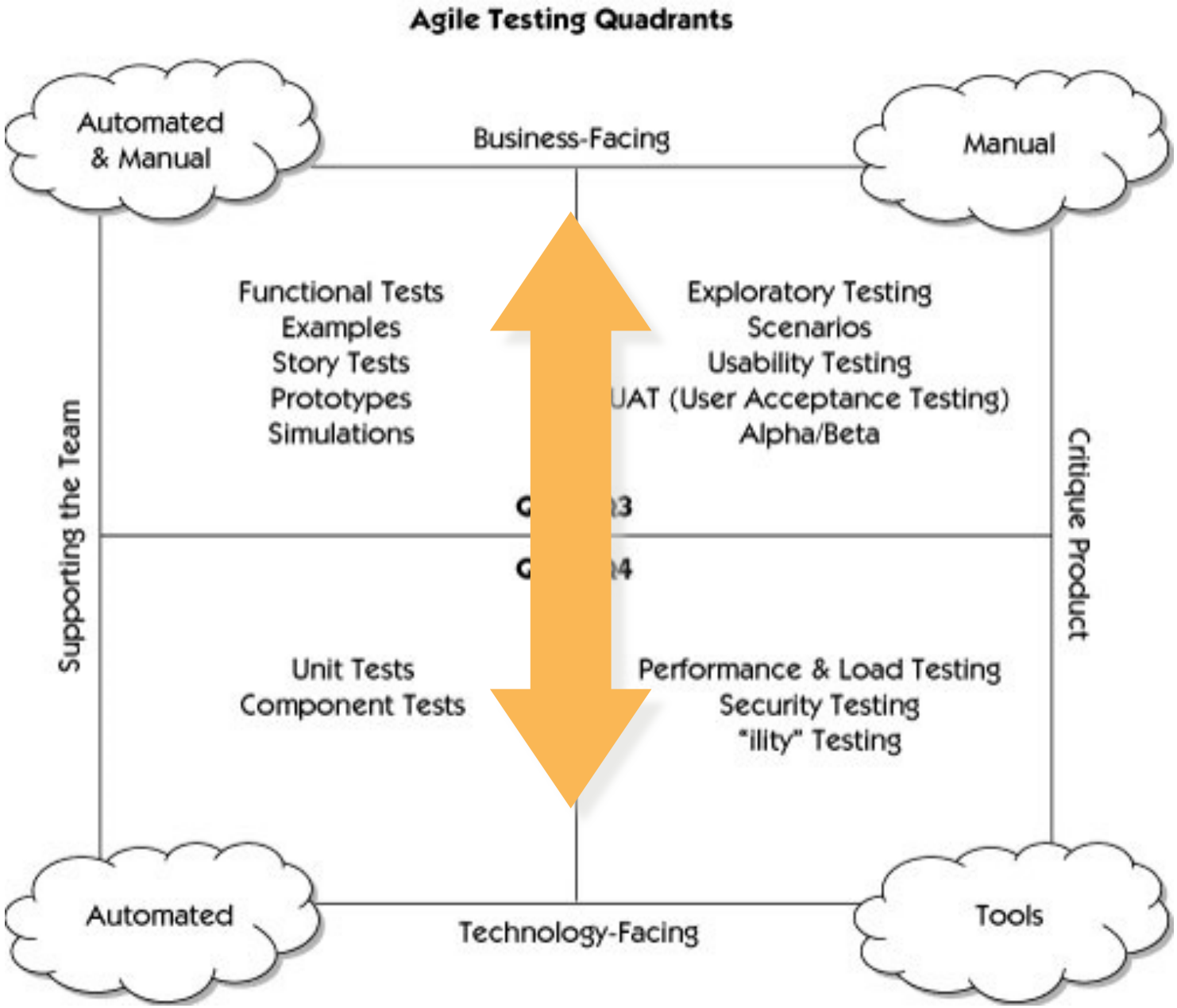
# At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

- Plan-D
  - On t
    and
  - For
    have
    work
    - Ca
  - On r
    - Ac
    - Do
      - then really doing something for it
      - Spoiled frequently by veto of “the chicken”

"When testing on Agile projects, distinguish between the challenges that are inherent in Agile methodologies from those challenges that arise from the improper implementation of Agile methodologies. The former challenges arise from too little tailoring of the chosen Agile methodology, while the latter challenges arise from too much tailoring." [Rex Black]

"While I think automated unit testing is the bees knees for a lot of reasons, the belief that testing is a straightforward, prescriptive business process that can be easily automated away - instead of an empirical investigative process ... well, sure, it's insulting, but more importantly, it's just well, wrong. " [Matt Heusser]

So how is it – Could all testing be Agile – or even agile?



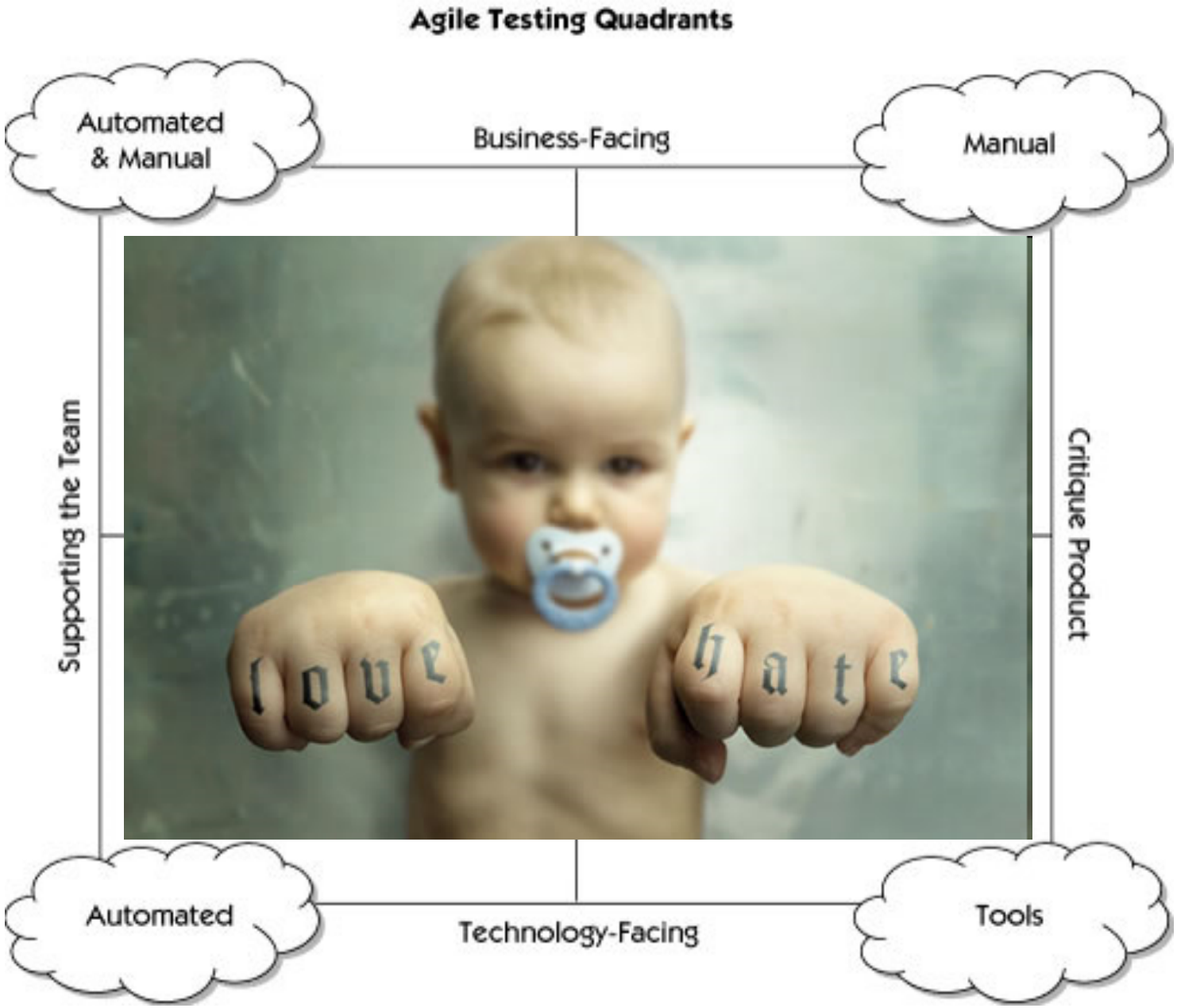

[Lisa Crispin and Janet Gregory from the idea of Brian Marick]

# The technology facing tests are tied to programming paradigm and tools

- “Tests that fall into programmer’s domain and are described using programmers terms and jargon” [Brian Marick]
- You can do TDD in plan-driven world, especially if you do continuous integration
- But you loose some of the benefits of practices
- Maintaining the momentum in “mixed” paradigm is less likely
- The definition of “Done” is a breath of fresh air for all testers soiled in waterfall death march projects, broken integrations and bad excuses
  - Allowing testers to concentrate on more creative aspects and into better communication

# The business-facing testing practices can survive on plan-driven world

- “Business-facing tests define the business experts’ desired features and functionality” [Crispin, Gregory]
- One key area where tester skills add value in agile team
- Exploratory testing fits into any paradigm
  - Great fit into agile via continuous learning, and allowing the learning affect the direction of testing
- “Any testing that fits the agile values and principles is agile testing” [Maaret Pyhäjärvi]
- Some of agile testing can live in plan-driven context, making testing more flexible
  - In general testing is expected to be more flexible and the rest of the project
  - “Testing as a service” sounds enticing




[Lisa Crispin and Janet Gregory from the idea of Brian Marick]

# Confirming or Critiquing

- Most unit testing efforts without proper coach will stay in confirmatory mode (yep, it's there) unless pushed a little to be more destructive
  - Breaking own code on purpose can be learned!
- Some customers can start aggressive out of suspicion but they can tame too easily
  - Can be solved with brainstorming and tester support
- Testers' natural attitude of constructive destruction & related skills is high currency even in Agile context
  - Changing hats takes very much energy and does not always work

# How agile is my plan-driven testing? Which way is it done now?

- Separate testers, unskilled labour to test based on detailed instructions
- Strict processes delivering documents to keep the herd of testers up-to-date on what should be verified
- Developers don't test
- Testing is scheduled to be "two weeks in the end", with long preparation without touching the software

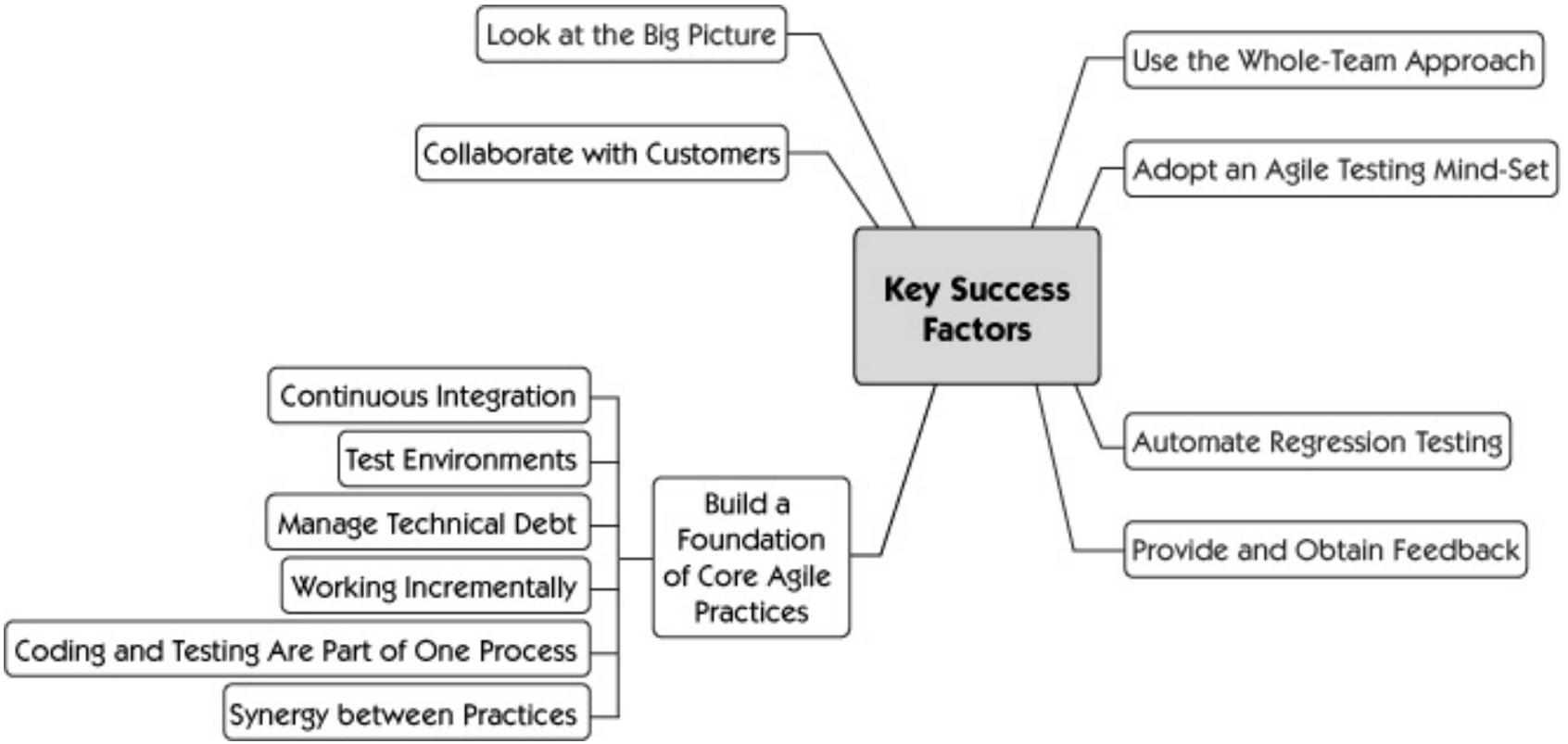
- Trained professionals with a different perspective than the developers
- Continuous collaboration with the whole development team and various perspectives to learn what to test
- Developers test and there's still work for separate tester
- Testing is continuous and how it is organized and resourced can vary a lot

***In which culture would you like to work as agile coach?***

# Bonus: Agile Testing Key Success Factors [Crispin, Gregory]

1. Use the whole team approach
2. Adopt an agile testing mindset
3. Automate regression testing
4. Provide and obtain feedback
5. Build a foundation of core agile practices
6. Collaborate with customers
7. Look at the big picture

- Continuous integration
- Test environments
- Manage technical debt
- Working incrementally
- Coding and testing are part of one process
- Synergy between practices




[Lisa Crispin and Janet Gregory]

tieto.com

# Thank you.

**Erkki Pöyhönen**
Business Development Manager
TietoEnator Corporation
Global Testing Practice
erkki.poyhonen@tieto.com