Automation testing in Agile projects - Overview

Shirly Ronen-Harel

Mar 2014

http://www.wired.com/insights/2013/04/big-data-fast-data-smart-data/

Who am I?

Linked-In: il.linkedin.com/pub/shirly-ronen-harel/0/653/249/

Twiiter : @ShirlyRonenRL

Email : shirly@agile-planet.com

Blog : Agile@home with your family and kinds
:http://agileandfamily.blogspot.co.il/

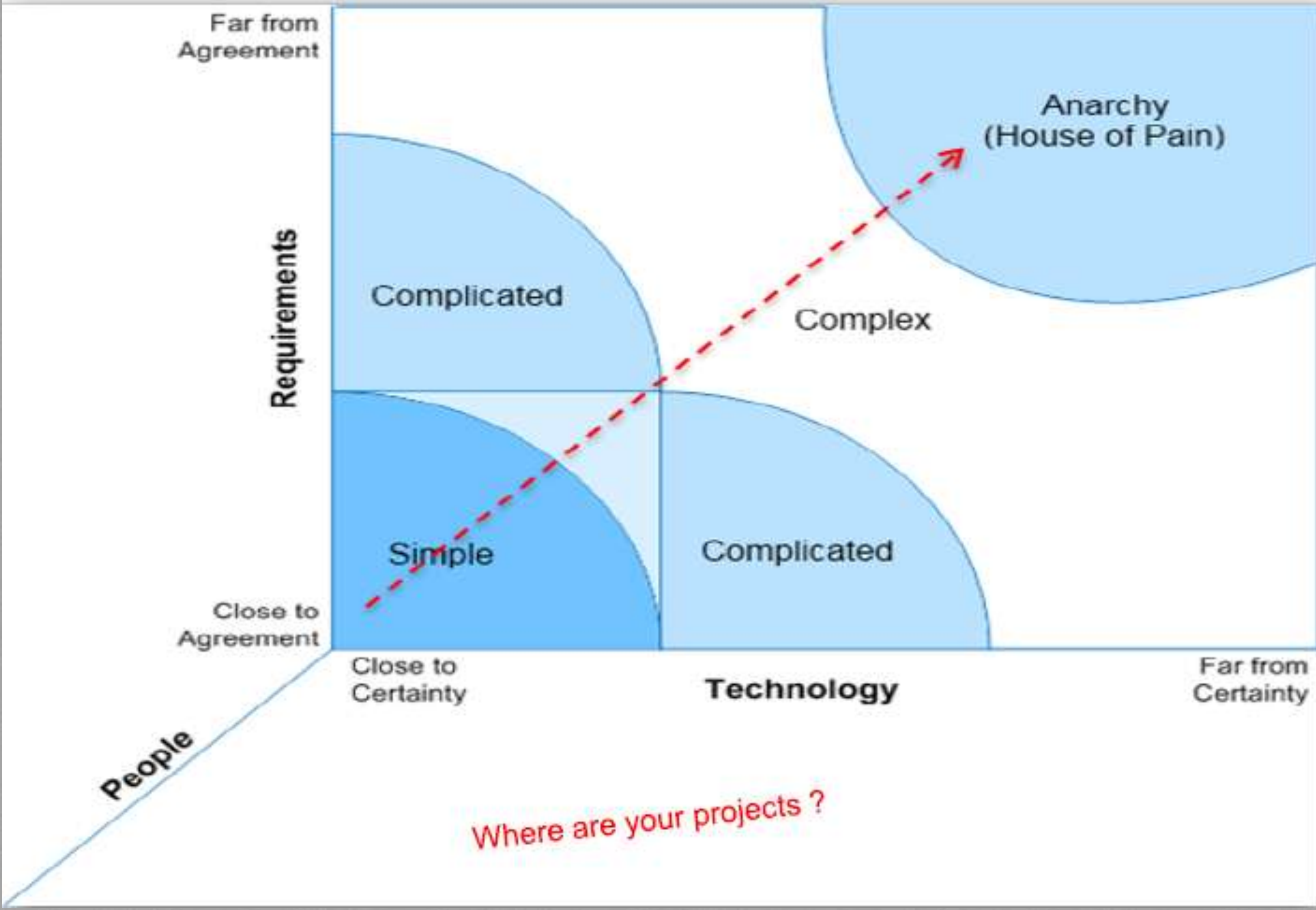
The Agile kids Book : http://agileandfamily.blogspot.co.il/p/agile-kids-book.html

Agilopedia : Everything about agile : @Roojoom :
http://agilopedia.blogspot.co.il/

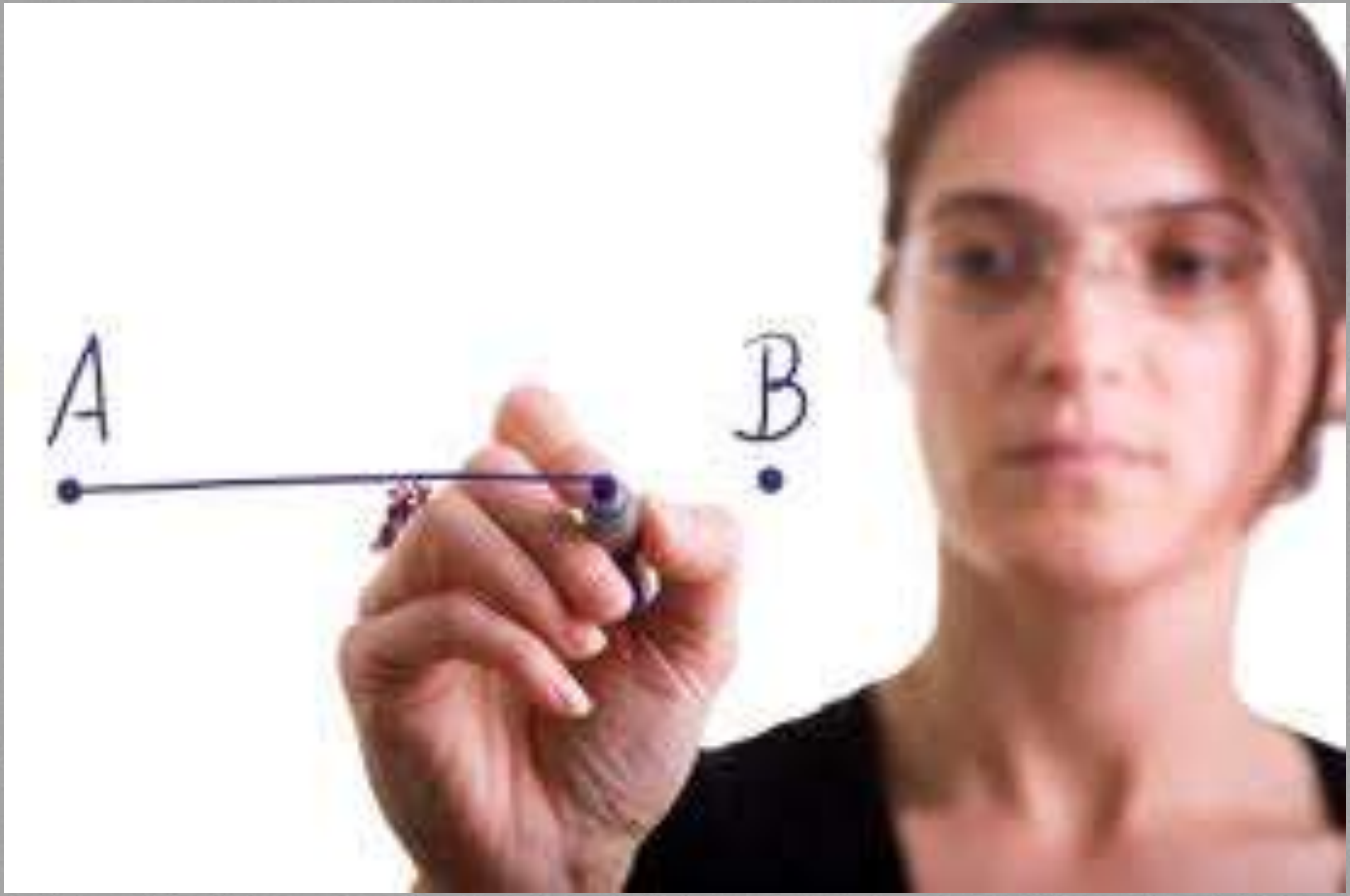
Takeaways

- The basics of Agile automation testing strategy
- Agile testing mindset
- Agile Testing quadrants and automation
- Agile testing project/release/iteration planning and execution (HL)

We live in uncertain times...

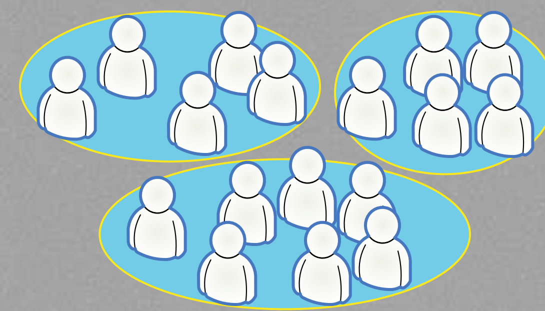


From A to B in the fastest way

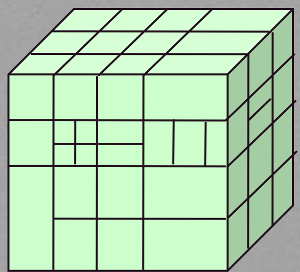


Scrum in a nutshell

Split your organization



Split your product

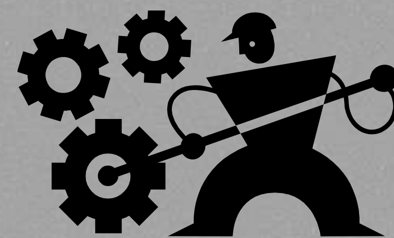


~~Large group spending a long time building a huge thing~~

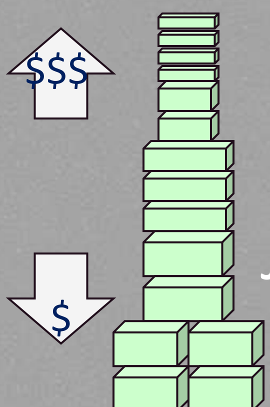
Small teams spending a little time building a small thing

... but integrating regularly to see the whole

Optimize process

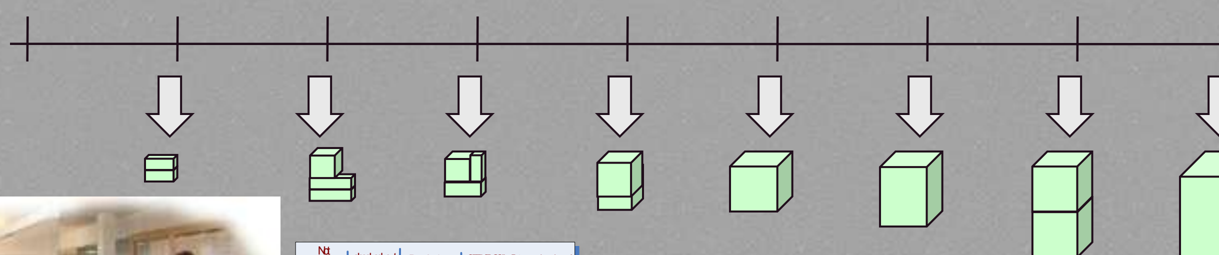


Optimize business value

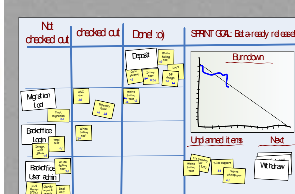


Split time

January

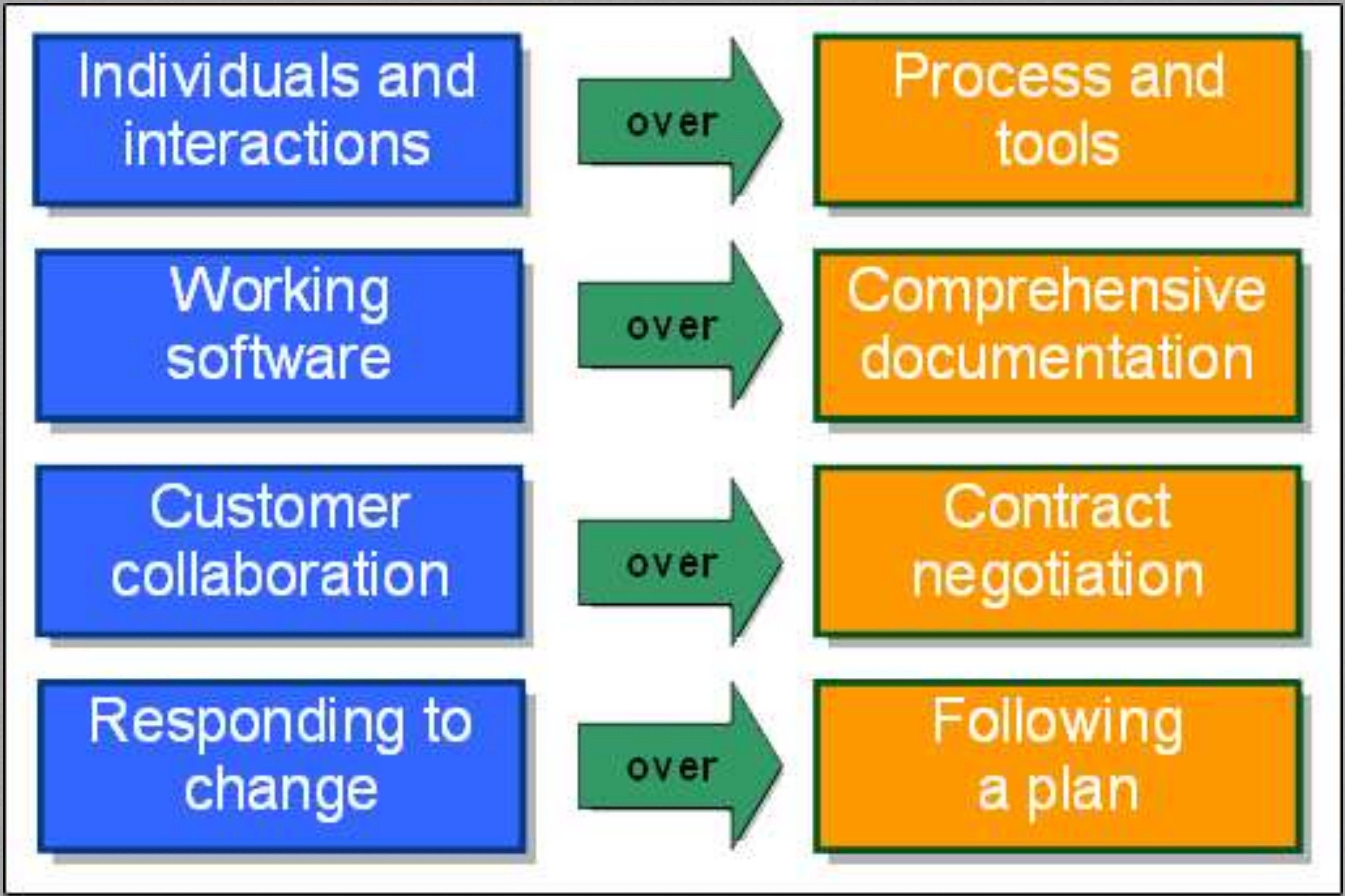


April



Agile aims to deliver something that can go live, in a relatively short period of time

Henrik Kniberg


Individuals and interactions
over
Process and tools
Working software
over
Comprehensive documentation
Customer collaboration
over
Contract negotiation
Responding to change
over
Following a plan

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

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

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

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

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

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

07 Working software is the primary measure of progress.

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

09 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.

Content

1

2

3

4

First up – the agile testing - reminder
The core of Agile testing
The testing quadrants
The testing team structure
The agile testing mindset

Traditional Tester’s Role

- Mainly a quality control function:
  - developing test plans against requirements
  - designing tests
  - executing tests
  - logging discovered defects
  - Providing some defects metrics
- In short, validating that the software works as specified

Testing goals

Testing is not a Phase !!!

Testing is the only way to approve that something is really done **from an end customer's perspective**

Testing provide information about the system that can help various stakeholders in their decision

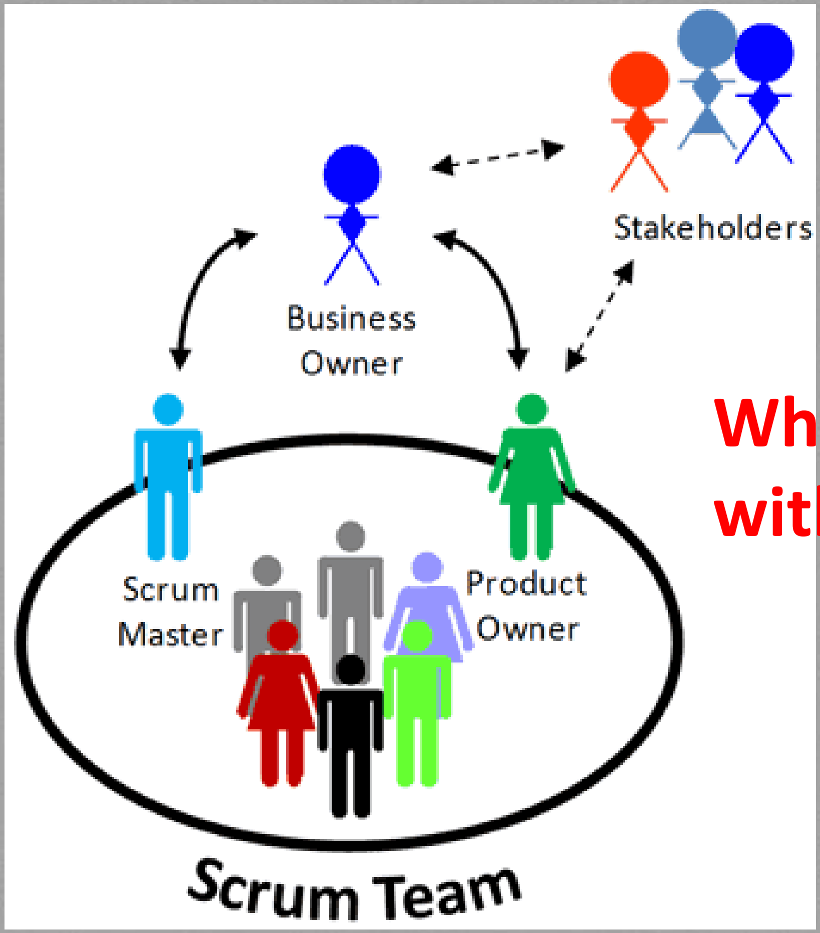
Why is there a deference from traditional testing and what is it?

http://www.howtogeek.com/howto/30941/whats-the-difference-between-jpg-png-and-gif/

The Tester’s Role in Agile

- Becomes the “right hand man” of the Product Owner
- Automation frees tester’s time to explore the business needs

- “Quality Assurance Champion” for the whole team
- “Headlight” of the team/project’s quality status
  - Continuously Improving the **Approach to Quality**



What should we do with the tester?

Testers move out from the QA group into the teams. Why?

Inspired by Henrik Kniberg ScruML http://blog.crisp.se/henrikkniberg/2007/08/25/1187995980000.html

Its not the tester that is part of the team – It’s the testing

The agile testing mindset

1. “We are all in this together ", “Whole team approach”, “collective testing ownership”, “A testing mindset is a team mindset.” “No more Us vs. Them. “

2. “ Development = Coding +Testing”, “We are building quality in “

3. “Continuous integration”, Don’t wait to the “integration hell period”

4. “Communication is a key to our success”.

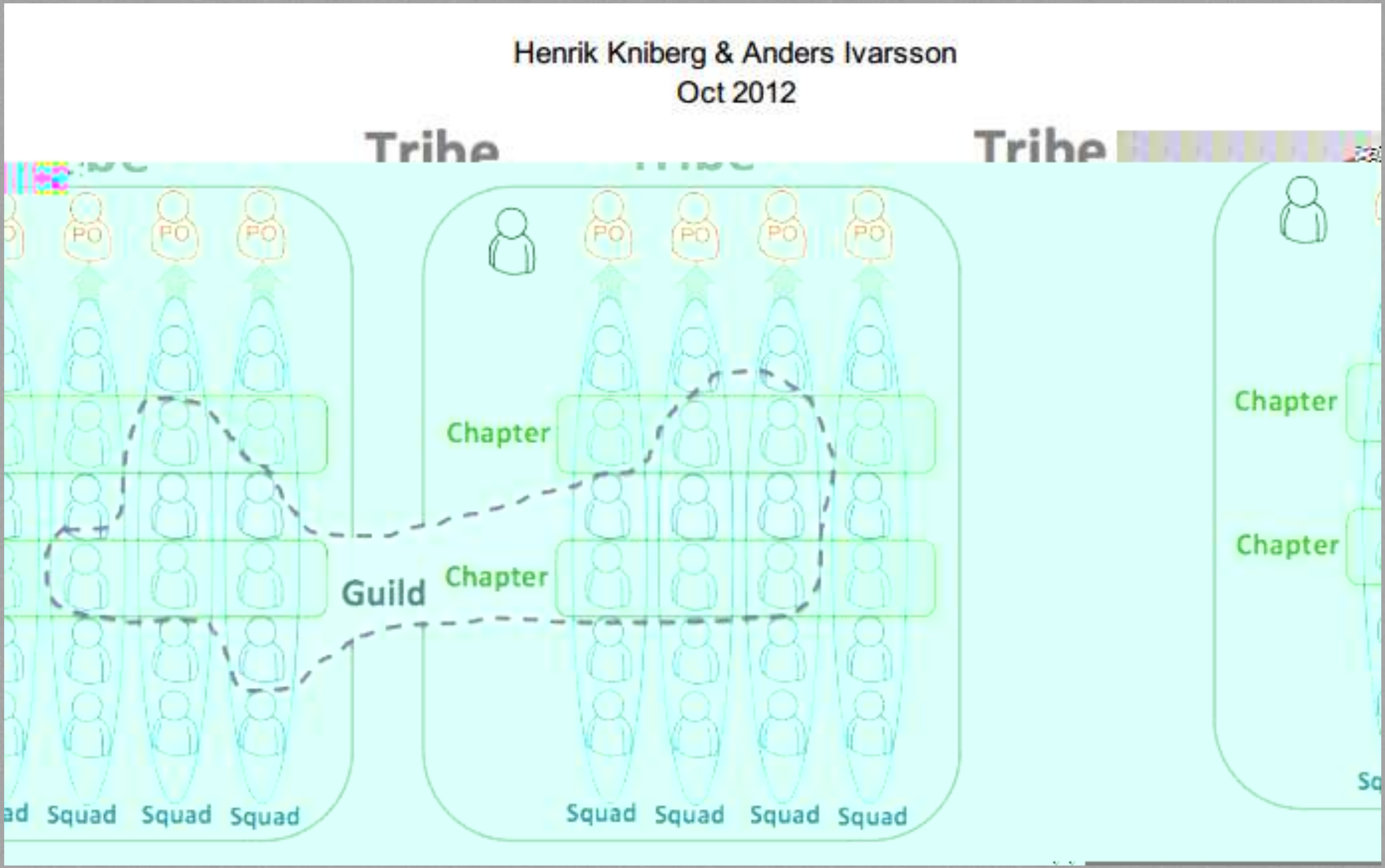
5. “Visibility across entire team members”

6. “Customer collaboration”. ”Testers are business oriented”

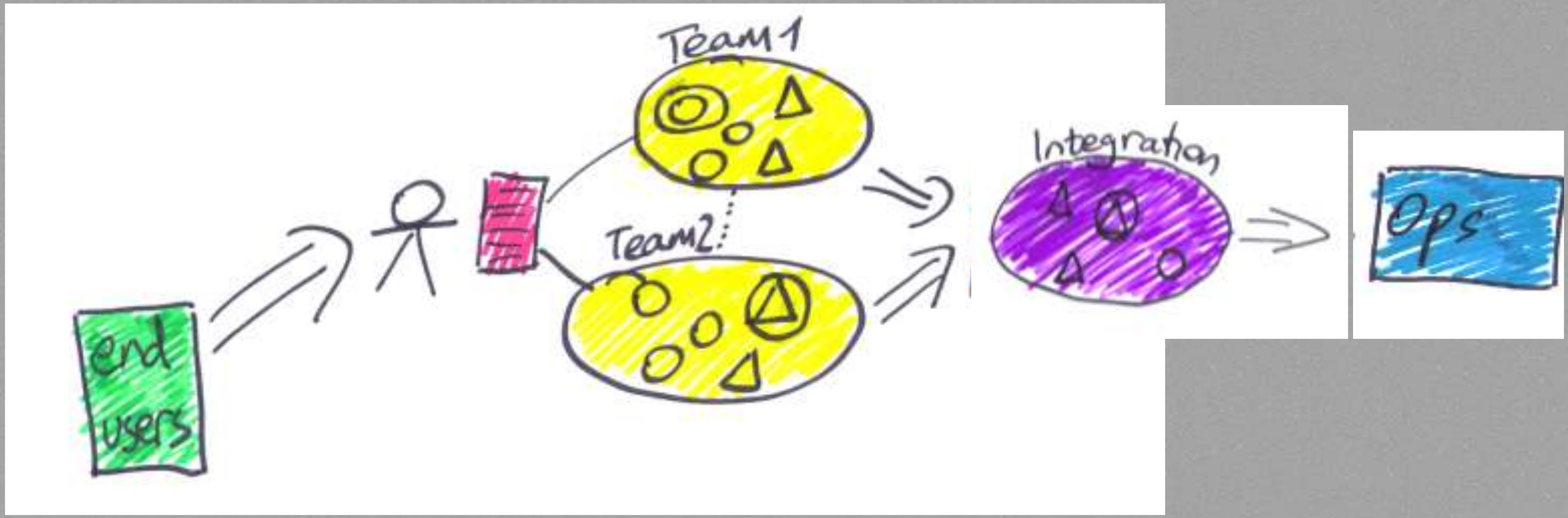
7. One of our goal is always working software (

8. Continuous improvement.

When we scale

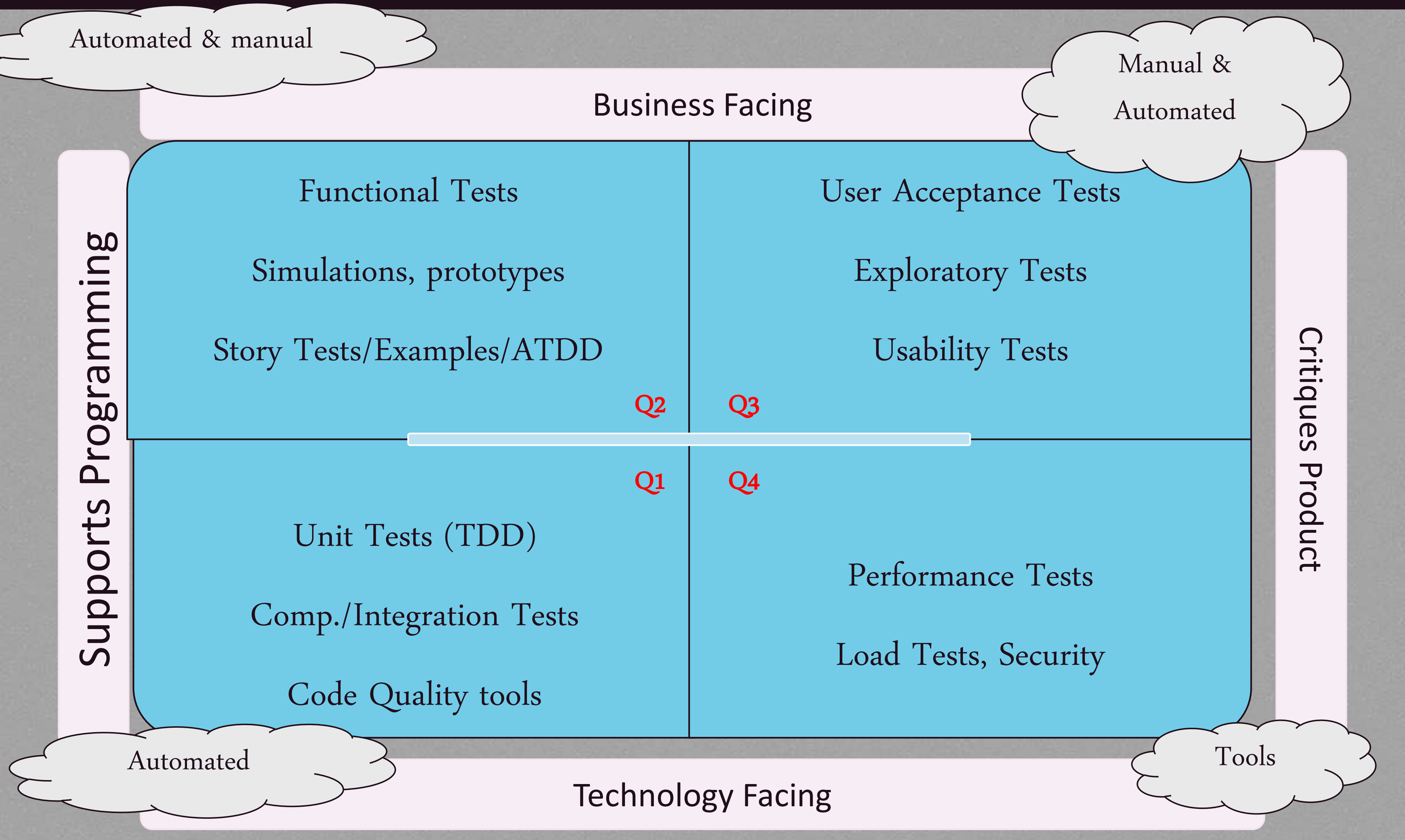


Independent integration team – …OPTIONAL



NOTE: Should only be done in very complex environments, where teams cannot handle real DONE DONE effectively

Brian Marick's Agile Testing Matrix



Content

1

2

3

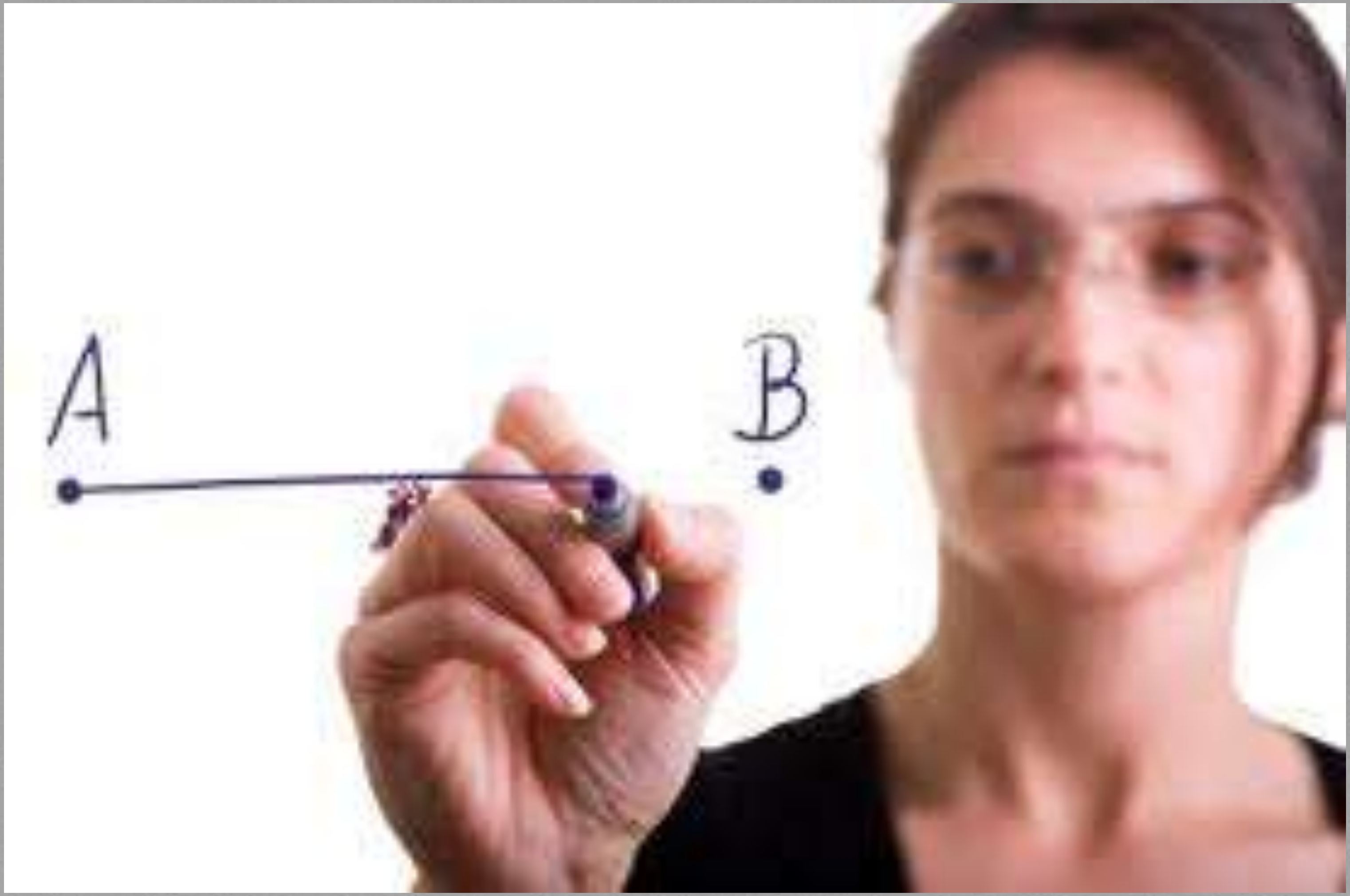
4

Automated testing in an agile world
The automation history
The automation strategy

Change And Fast

Early feedback (Why?)

From A to B in the fastest way



What are the Objectives for Automation ?

- Free up time for more important tasks
- Increase productivity and time to market by:
  - Reducing code freeze periods
  - Providing early feedback for quality issues
  - Repeatability
  - Fix fasts (closer to coding reality)
- Document behavior = executable specs
- Ensure reliable system at any given moment
- Feel safe to promote changes into production

Typical Automation – Top Down

Record & Play → **DDT Data Driven Testing** → **KDT keyword driven test** → **Hybrid** → **Agile Testing**

Record & Play: Very sensitive, only suitable for stable system late in the process, require manual effort, Slow execution, inflexible and not easily extended

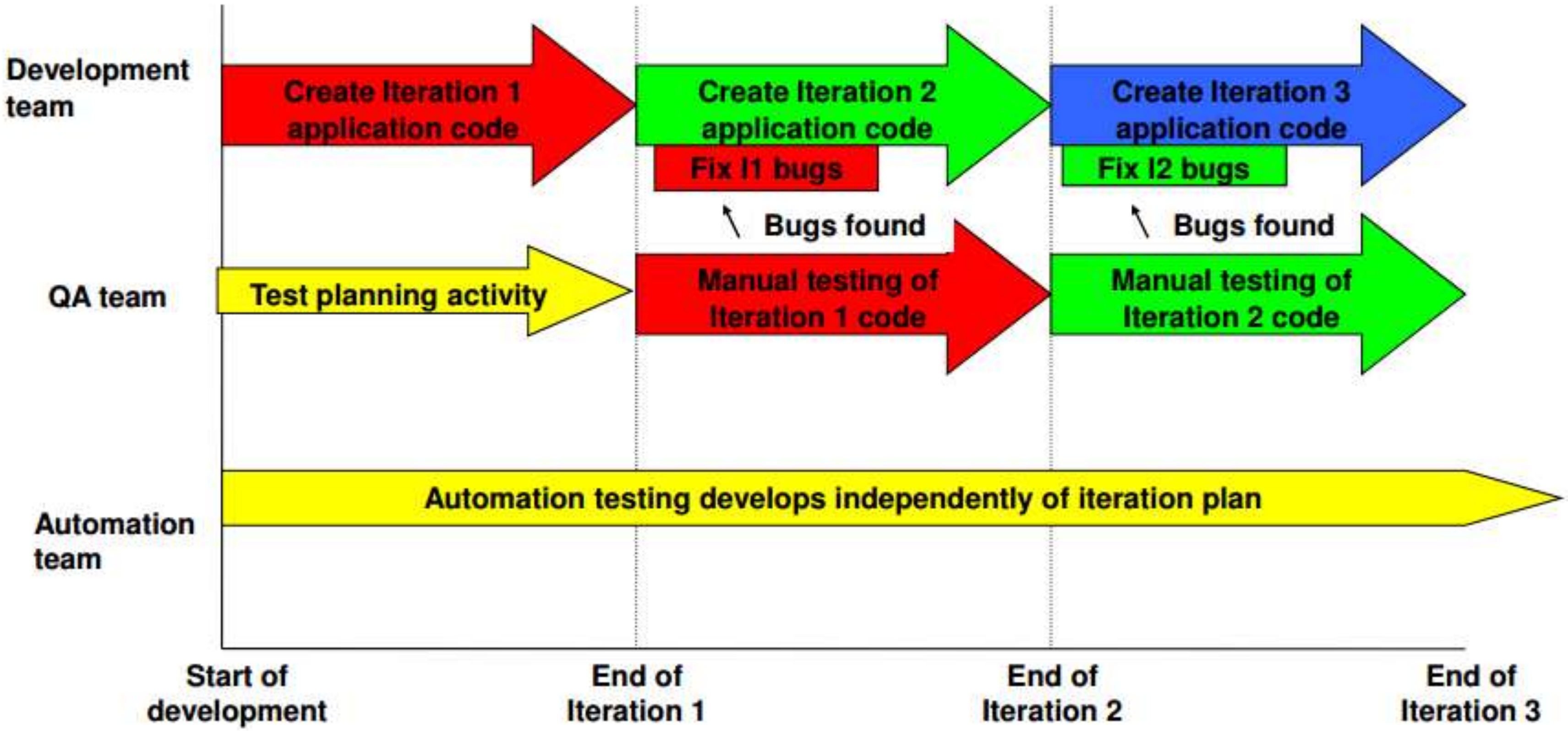
DDT: Data that is external to functional tests. Tests may be altered by easy (excel) introduction of variables to infinitely extend test cases (simply adding new lines)

KDT – a software testing technique that separates the programming work from test design. Consists of keywords (business functions and UI operations), and business scenarios (login, sell item,…). Known as very powerful approach to achieve more test automation, early in solution life-cycle, at low maintenance

Hybrid: Modular testing which involves DDT and KDT, achieving a higher coverage automated test solution.

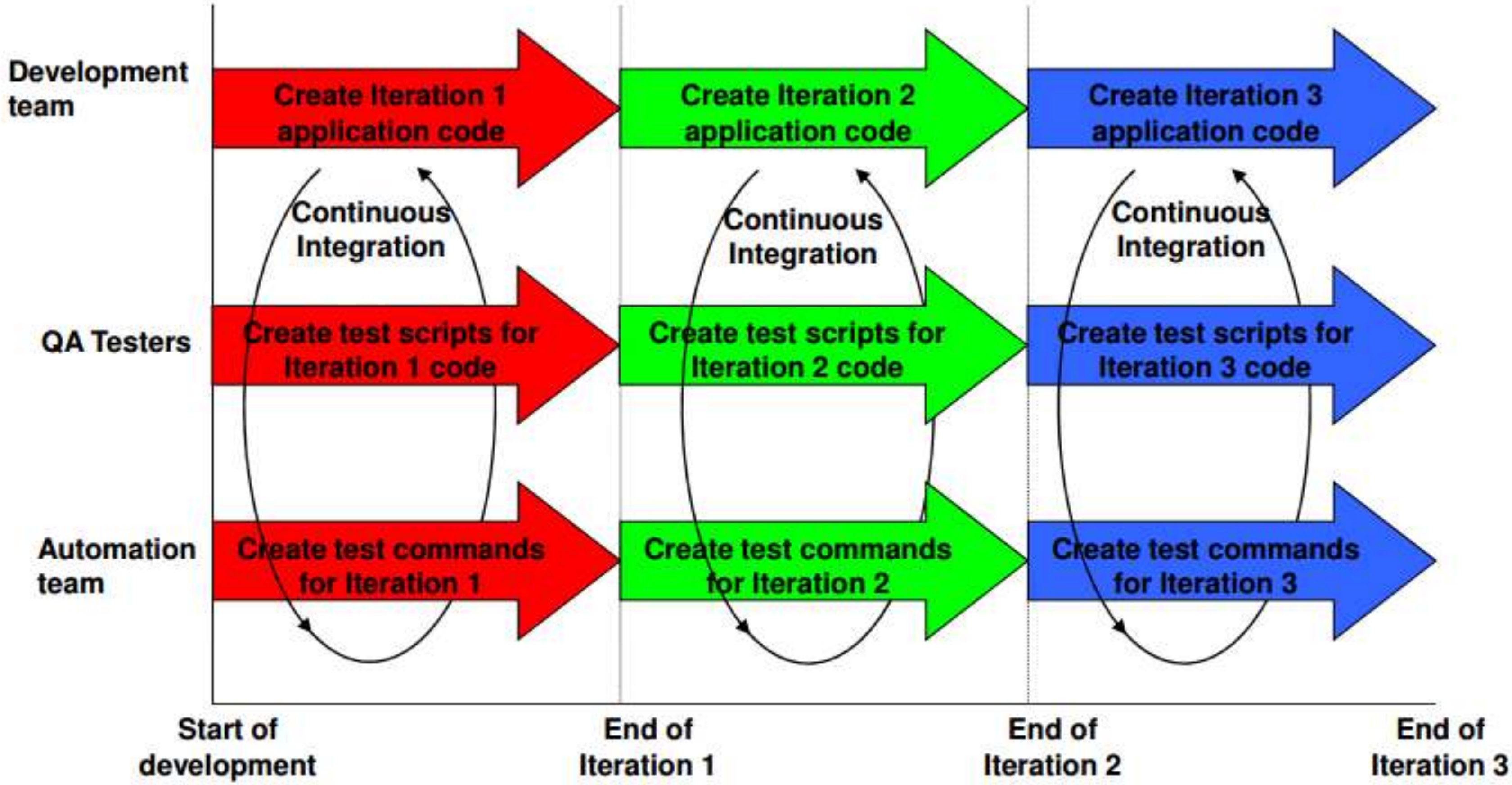
Agile testing: a combination of ATDD (acceptance test driven development) and embedded unit testing. Introducing a continuous integration testing approach to increase confidence in solution in product and solution layers.

Old way



Development team
Create Iteration 1 application code
Create Iteration 2 application code
Create Iteration 3 application code
Fix I1 bugs
Fix I2 bugs
Bugs found
Bugs found
QA team
Test planning activity
Manual testing of Iteration 1 code
Manual testing of Iteration 2 code
Automation testing develops independently of iteration plan
Automation team
Start of development
End of Iteration 1
End of Iteration 2
End of Iteration 3

New way



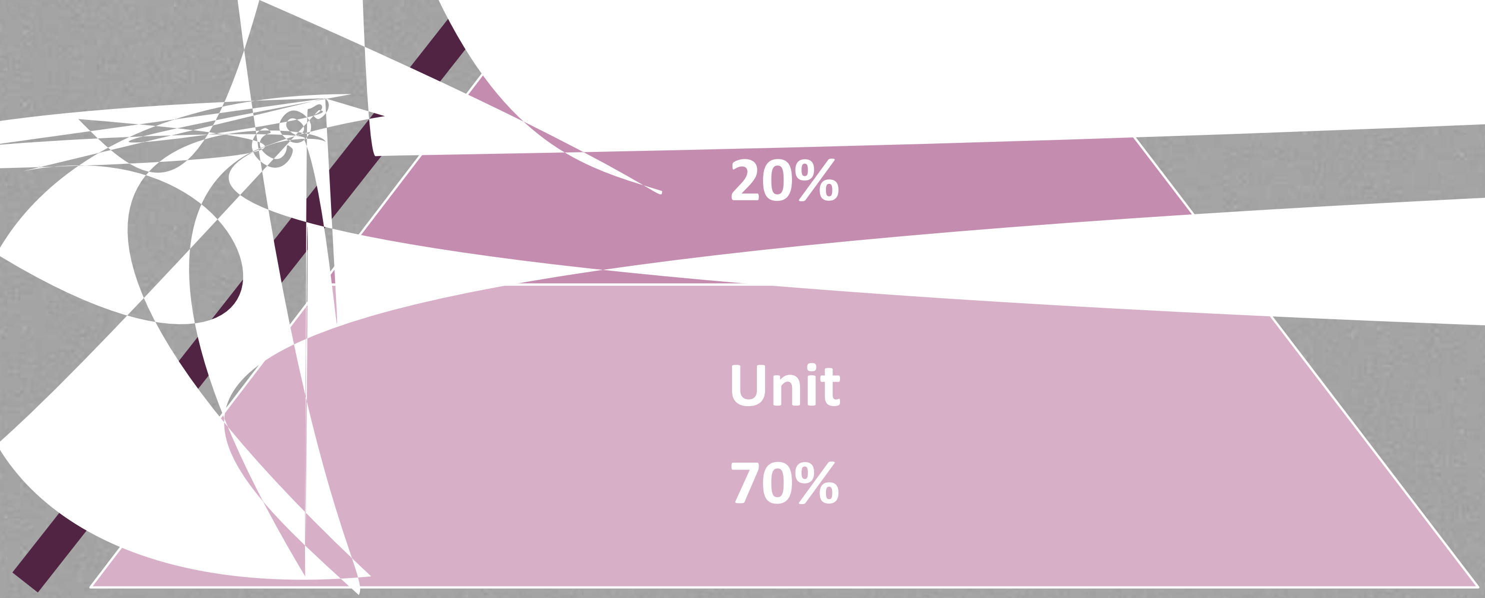
Development team
QA Testers
Automation team
Create Iteration 1 application code
Create Iteration 2 application code
Create Iteration 3 application code
Continuous Integration
Continuous Integration
Continuous Integration
Create test scripts for Iteration 1 code
Create test scripts for Iteration 2 code
Create test scripts for Iteration 3 code
Create test commands for Iteration 1
Create test commands for Iteration 2
Create test commands for Iteration 3
Start of development
End of Iteration 1
End of Iteration 2
End of Iteration 3

Shared ownership

Test Automation Pyramid

Record and Play → DDT → KDT → Hybrid → Agile Testing

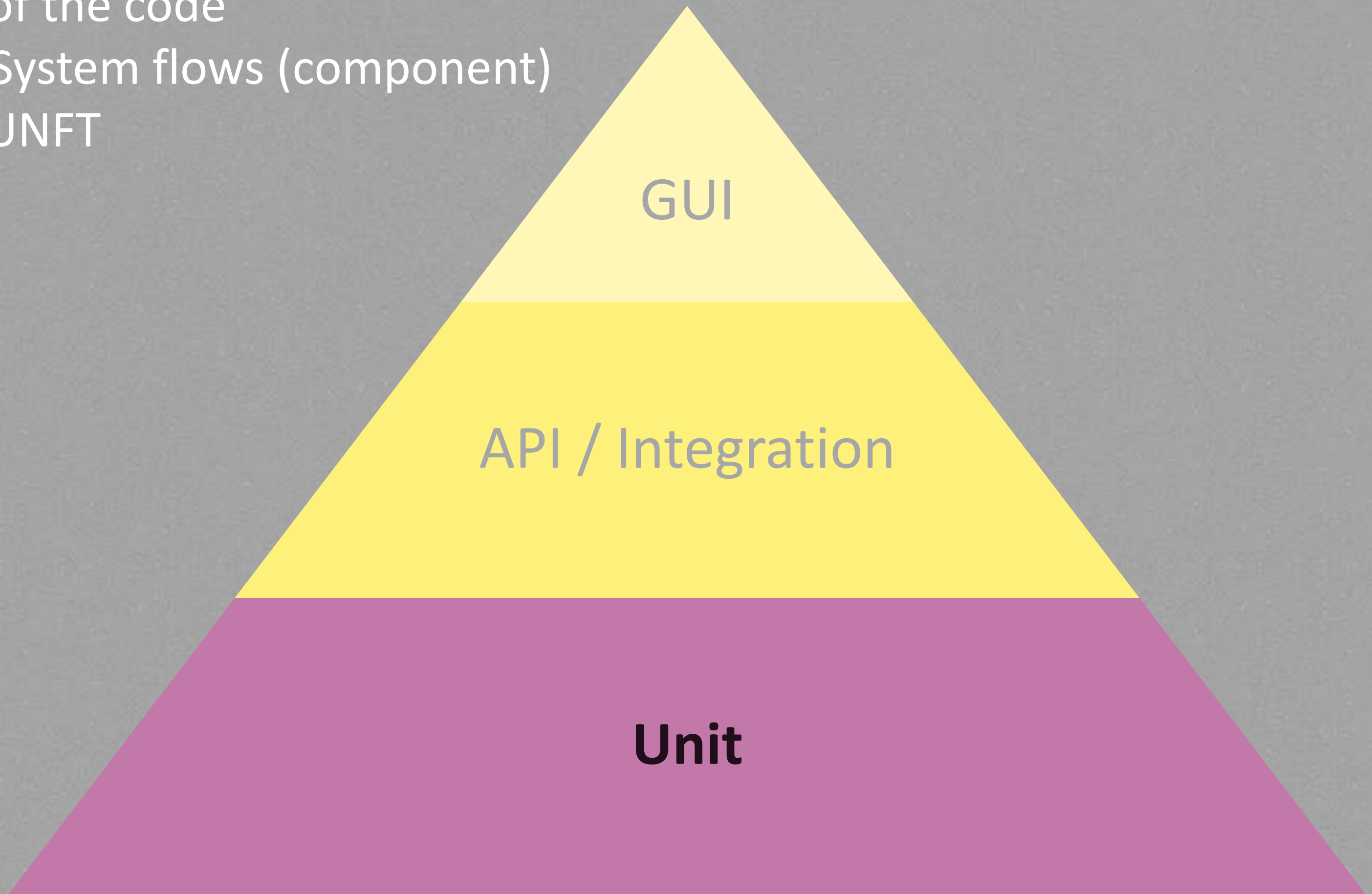
T tom ti



Test Automation Pyramid

Unit

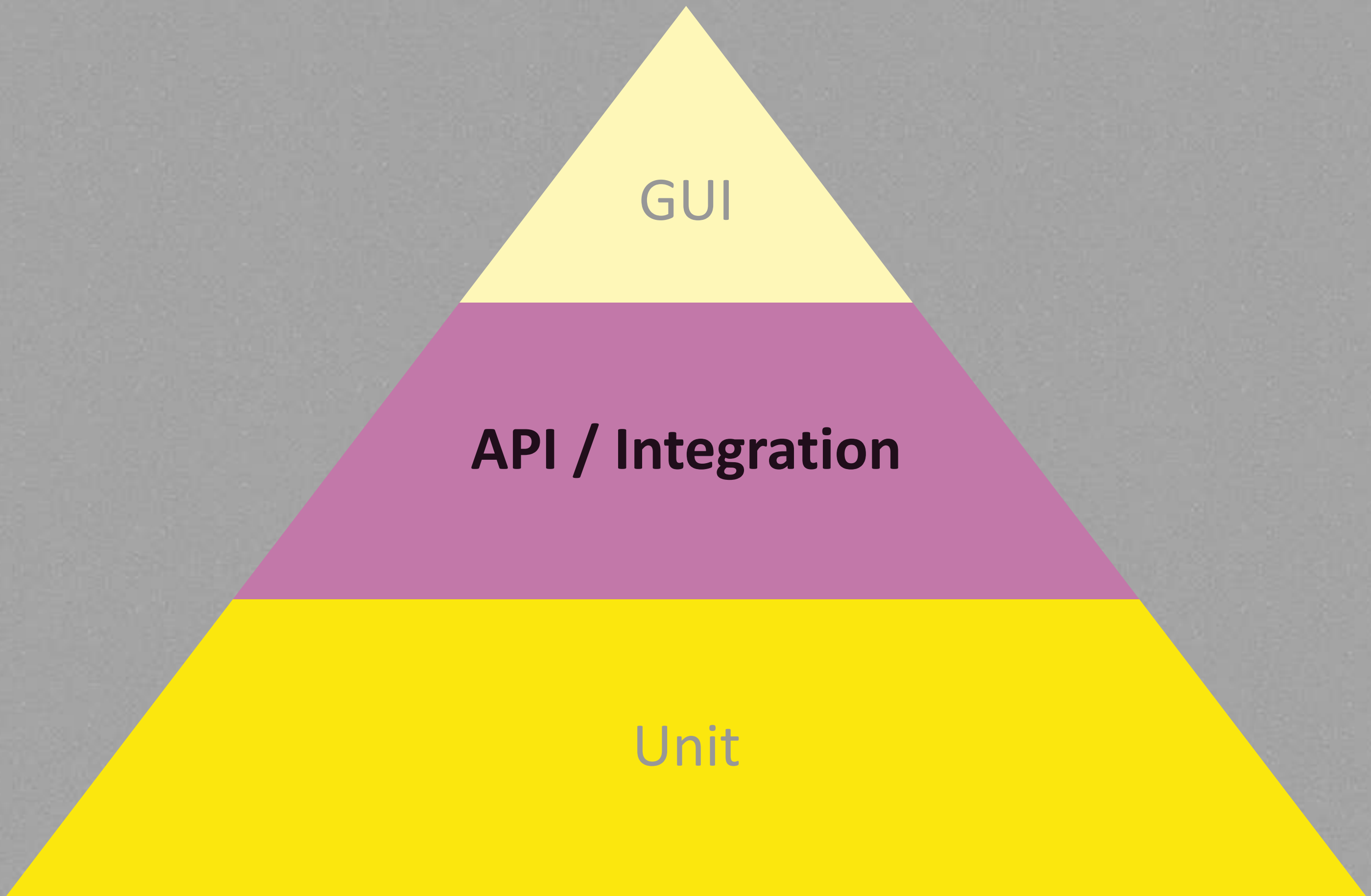
- Foundation
- System language
- Fast feedback
- Part of the code
- Also System flows (component) and UNFT



Test Automation Pyramid

API / Integration

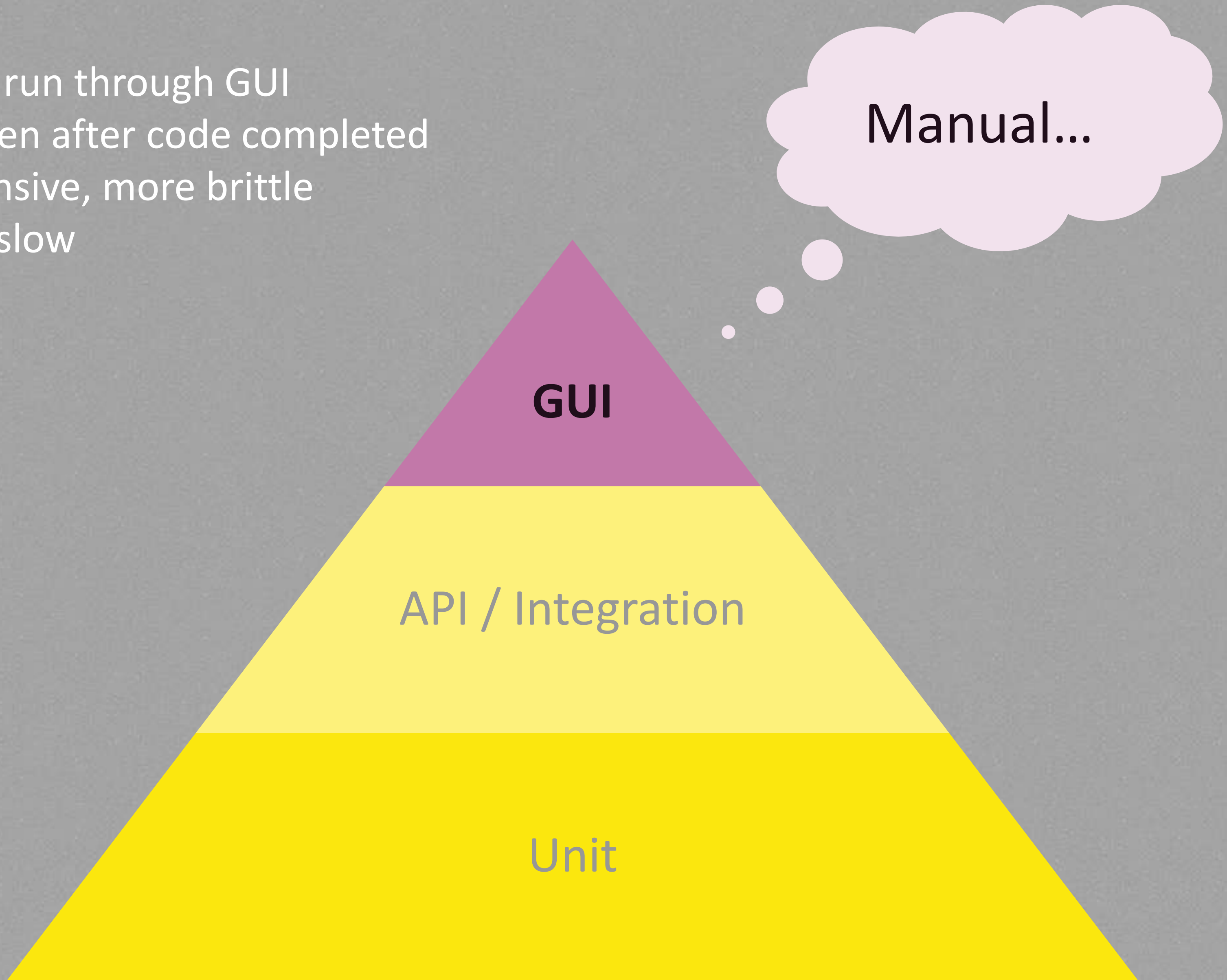
- Business logic behind the GUI,
- Understood by customer !!!
- Slower (DB, wider scope)



Test Automation Pyramid

GUI

- Tests run through GUI
- Written after code completed
- Expensive, more brittle
- Very slow



Automation is part of DOD

What shouldn’t we automate?

Always think what Shouldn’t we do as part of our discovery and planning continues process

- Push automation down the pyramid
- Repeatable and self checking
- Aim to low maintainability of tests (re-use, pointers to tests, stand alone…)
- Maintainable - Re-factor tests
- Specific
- Avoid DB access when applicable
- Use source control for tests
- Interrupt in code design for better testability of code

Tips

- Start small
- Create a test automation backlog
- Keeping your tests and data loosely bound will help you to switch testing tools with ease in the future if such a need arises.
- Create meaningful tests
- Keep it green -Make every effort to resolve test failures quickly and keep test execution times as short as possible.

Tips

- Combine Automation Championship with shared ownership for achieving automation by DEV/Test
- Promote Automation as native part of the development and testing work
- Create meaningful tests and ensure that they don't create a false sense of security.
- Expand Automation scope to "Any use of tools to support all aspect of testing" [James Bach]
- Build Quality in
- Have an intuitive reporting mechanism in place
- and give everyone on the team visibility to test results and historical trends. This will help everyone involved in the project monitor the progress and health of development and make more informed decisions.

What are the steps? Agile Testing Matrix

Business Facing

	Supports Programming	Critiques Product
Business Facing	Functional Tests Simulations, prototypes Story Tests/Examples/ATDD	User Acceptance Tests Exploratory Tests Usability Tests
Technology Facing	Unit Tests (TDD) Comp./Integration Tests Code Quality tools	Performance Tests Load Tests, Security

Technology Facing

Content– Optional

1

2

3

4

Detailed automation
TDD (Test driven development)
ATDD (Acceptance TDD)
BDD (Behavior…)
Acceptance
CI (Continues integration)
CD’s (Continues delivery, deploy…)

What are Acceptance test?
Exercise

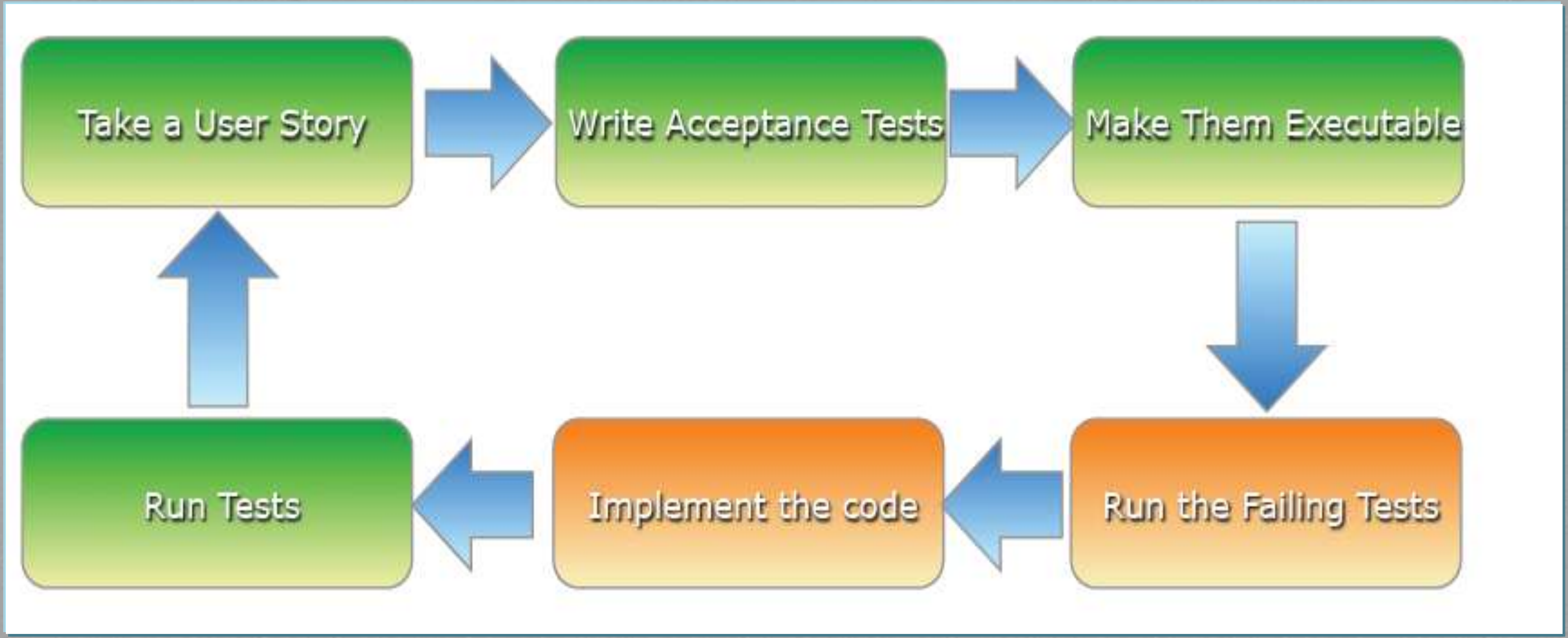
TDD & ATDD

- ATDD - Acceptance Test Driven Development – A Planning, Documentation and Testing Methodology (business level)
  - **Turning user stories and expected behavior into acceptance tests**
  - Preferably those tests will eventually become automatic

- TDD - Test Driven Development, a development methodology for delivering high-quality **code** (technical level)

The ATDD Cycle

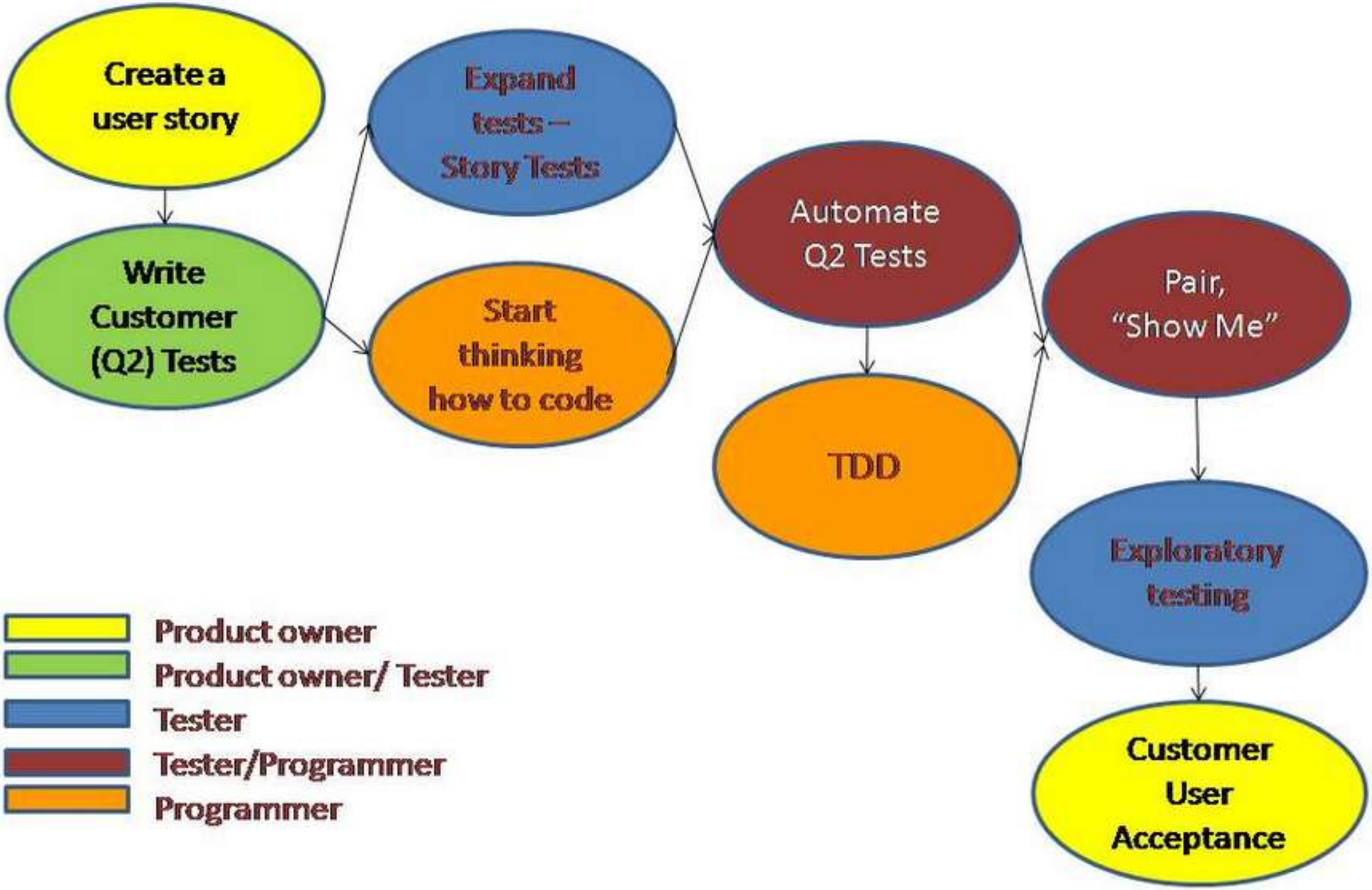
- ATDD makes sure the user story is completed end to end
- TDD makes sure the different units are implemented with high quality code (OPTIONAL)



Principles

- Working software over comprehensive requirements
- Executable spec

ATDD (Acceptance Test Driven Dev)



BDD

BDD focuses on the behaviors of your system exhibits than the implementation details of it.

http://erpleads.blogspot.co.il/2012/09/how-b2b-appointment-setting-responds-to.html

BDD

- Obtaining a clear understanding of desired software behavior through discussion.
- It extends TDD by writing test cases in a natural language that non-programmers can read.
- Using examples
- Allows the developers to focus on **"Does my code do what it's supposed to do?" in addition to "Does my code work?"**

BDD structure (Given-When-Then)

[**Feature/MMF** title]As a [role] I want [feature] So that [benefit]Scenario 1:

[**Title/test purpose**]

Given [context/system state]And [some more context]...

When [event/action] and [next event/action]

Then [outcome]and [another outcome]...Scenario

BDD defines story as “ready story”

Team commits to BDD
PO writes User Story
QA prepares BDD
PO reviews with the QA & Dev and agree over what to test

QA develops the automation
Code review with developers
Tests run on dedicated environment
Once story is done – all ATDD Pass – the story may be integrated to other environments

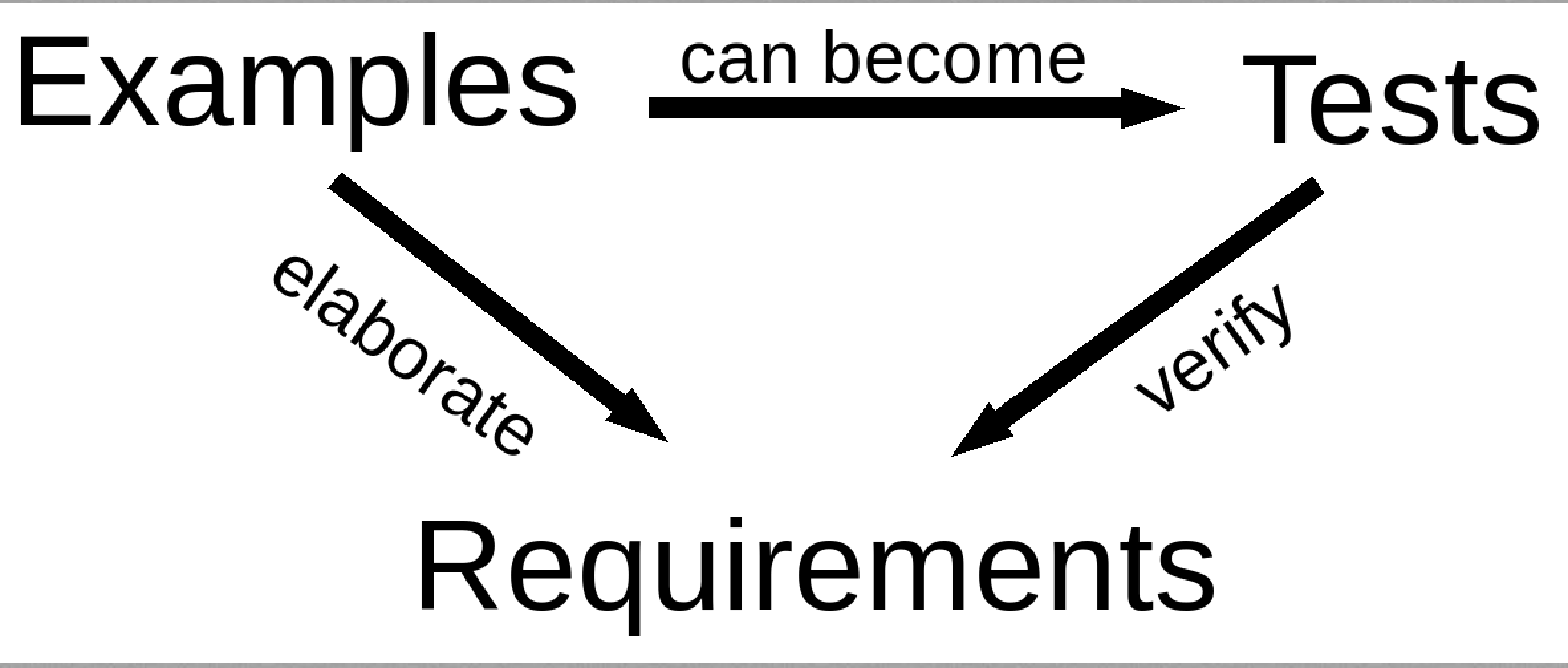
Specification By Examples

http://www.ivillage.ca/health/fitness/the-o-course-crucible-military-designed-race-tests-mind-and-body

What is specification By Example?

- Thin slices of system behaviour
- that deliver business value
- described as concrete examples
- that are potentially automatable
- without translation
- to create executable specifications
- captured in live documentation.

http://www.slideshare.net/dwhelan/specification-by-example-12983304


Examples
can become
Tests
elaborate
verify
Requirements

Specification By Example

Source: https://docs.google.com/drawings/d/1cbfKq-KazckMVCnRflh6zMSD8dtf9OKyiV7DoxGyWM/edit

Derive the Scope: User Stories

As a ________
I want to ________
So that ________

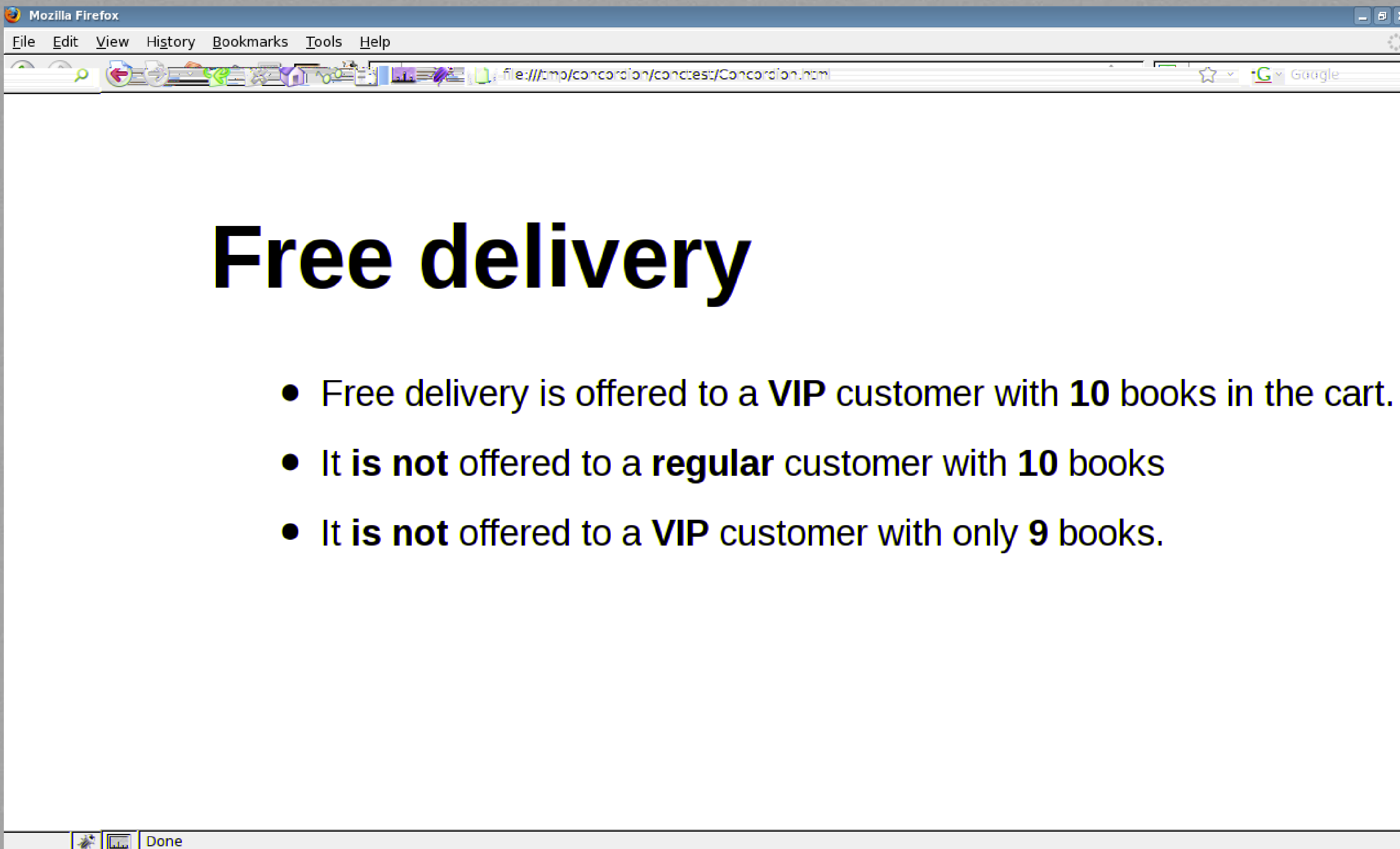
As a student
I want to purchase used books online
So that I can save money

Specify Collaboratively: Key Examples

Given ________
When ________
Then ________

Given "War and Peace" is available as a used book for $2.99
When Susan selects book "War and Peace"
Then "Buy used for $2.99" is displayed

This is a specification:



Free delivery

- Free delivery is offered to a **VIP** customer with **10** books in the cart.
- It **is not** offered to a **regular** customer with **10** books
- It **is not** offered to a **VIP** customer with only **9** books.

It is also a test

Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///tmp/concordion/conctest/Concordion.html

expresivenes

Free delivery

- Free delivery is offered to a **VIP** customer with **10** books in the cart.
- It **is not** offered to a **regular** customer with **10** books
- It **~~is not~~ is** offered to a **VIP** customer with only **9** books.

Results generated by concordion™

in 130 ms on 21-Aug-2008 at 01:47:23 BST

Done

This as well

The prize pool is divided among the winners using the following distribution for winning combinations (number of correct hits out of six chosen numbers). Example below is for $2M payout pool.

Prize Distribution for Payout Pool	2,000,000	
Winning Combination	Pool Percentage?	Prize Pool?
6	68	1,360,000
5	10	200,000
4	10	200,000
3	12	240,000

WHEN THERE IS NOT ENOUGH MONEY IN THE ACCOUNT, THE TICKET PURCHASE SHOULD BE REFUSED. THE TICKET SHOULD

not be registered, account balance and pool value remain untouched.

Player	john	Deposits	50	dollars with card	4111111111111111

Reject	Player	john	purchases a ticket for	100	dollars

Check	Pool value for draw on	01/01/2008	is	0

Check	Account balance for	john	50

Check	Tickets in draw on	01/01/2008	0

And this

- **Given** a stock of prices 0.5,1.0
- **When** the stock is traded at 2.0
- **Then** the alert status should be OFF
- **When** the stock is traded at 5.0
- **Then** the alert status should be OFF
- **When** the stock is traded at 11.0
- **Then** the alert status should be OFF

A good acceptance test is

- Focused on a single thing (rule, step..)
- Specification not a script
- Self-explanatory
- Uses the domain language
- SMART
  - Specific
  - Measurable
  - Achievable
  - Relevant
  - Time-bound

Automate it!

Developers will have to code exactly what was specified

… not just the rules they interpret

Or… Use the right tool

What will we gain out of it?

- Documentation
- PO
- DEV
- Testers

NFT (Non Function Testing)

Agile Non-Functional Testing

- Collective ownership of Non-functional behavior of the system
- **Perform NFT as early as possible = early feedback**
- Apply NF requirements as part of DoD of Features
  - **NF being part of Done Done!**
- New NF requirements can be a Feature/Story

Continuous automated testing – what types

Continuous integration

Continuous delivery and

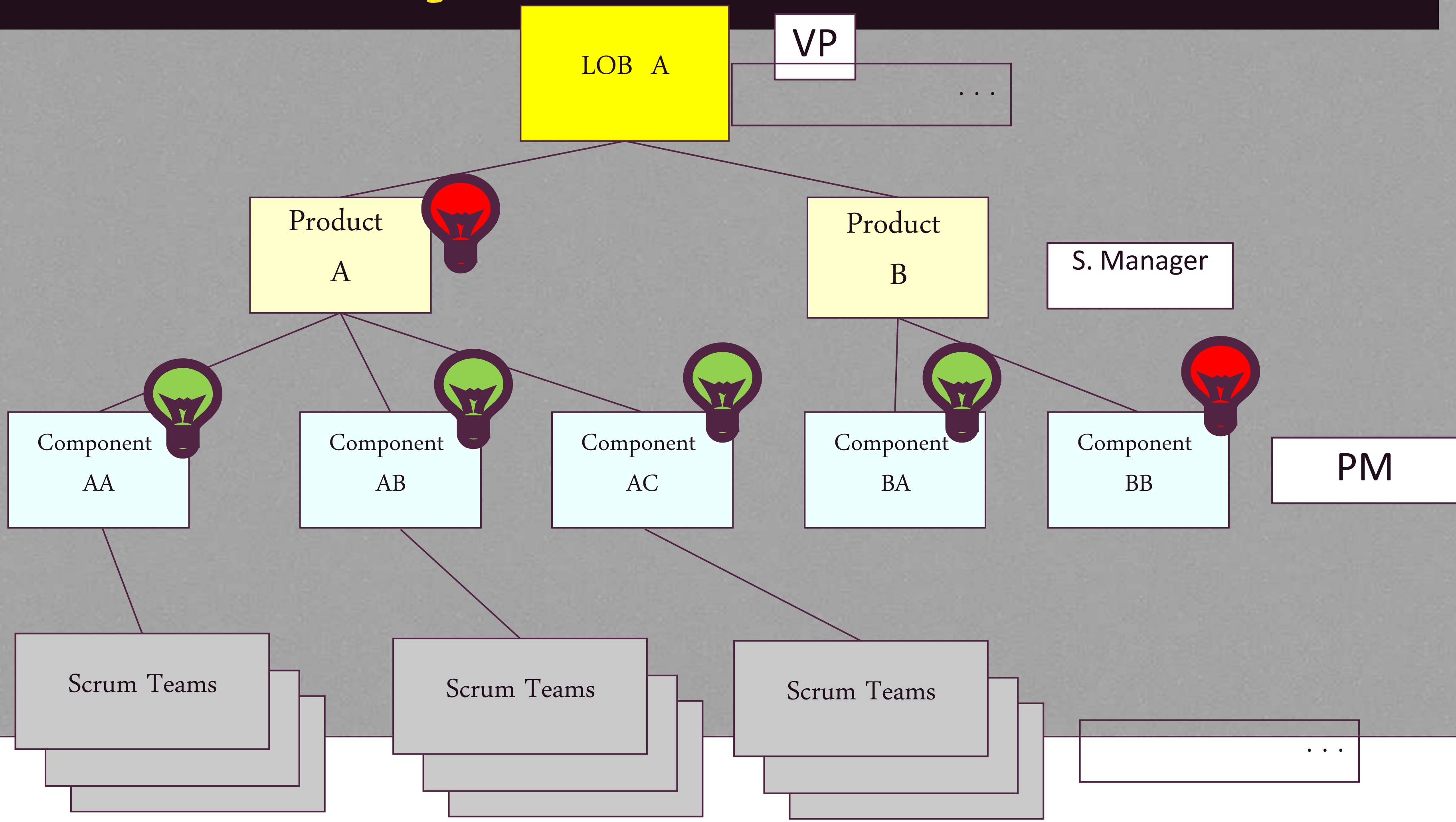
Continuous deployment

Continuous Integration (CI)

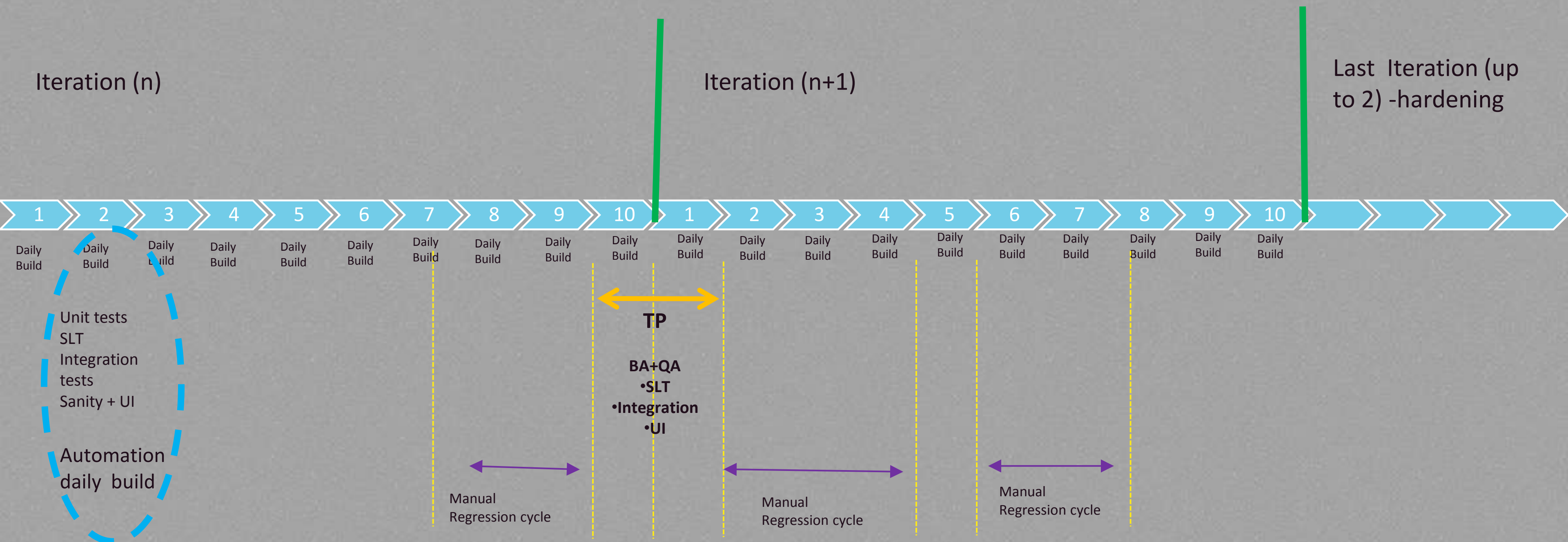
- Integrating early and often
  - Avoid "integration hell"
- Always have a **working software** based on latest code

- Using :
  - High automation coverage at low level
  - Close collaboration between development and testing
  - CI tool and framework
  - **“stop and fix” (Jidoka)**

Visibility / Build Dashboard



Example for Release Activities



Automatic Test Types example

	Test Subject	Tool	Who	Frequent	Measurements
Unit Test	Every piece of code		Development	Every Build	Per iteration - graph of how many UT were added. Success rate per day. Code coverage.
Service Level Test	Business Components	Tool XXX	Development	Every Build	Definition of done for a user story is to have SLT. Success rate per day.
Integration Test	Integrated system| (Server, DB etc). Testing tools acts as a client	Tool XXX	QA	Daily	For each functional requirement have E2E end flows that covers it. Measure Success rate.
End 2 End	Integrated system against live client	Mercury/ Manual	QA	Daily / Weekly	List of end to end flows and covered scenarios results Measure Success rate.
PerformanceTe sting	Integrated system (Server, DB etc). Testing tools acts as a client	Load Tools	QA	Daily / Weekly	List of end to end flows and covered scenarios results

Content – optional?!

1

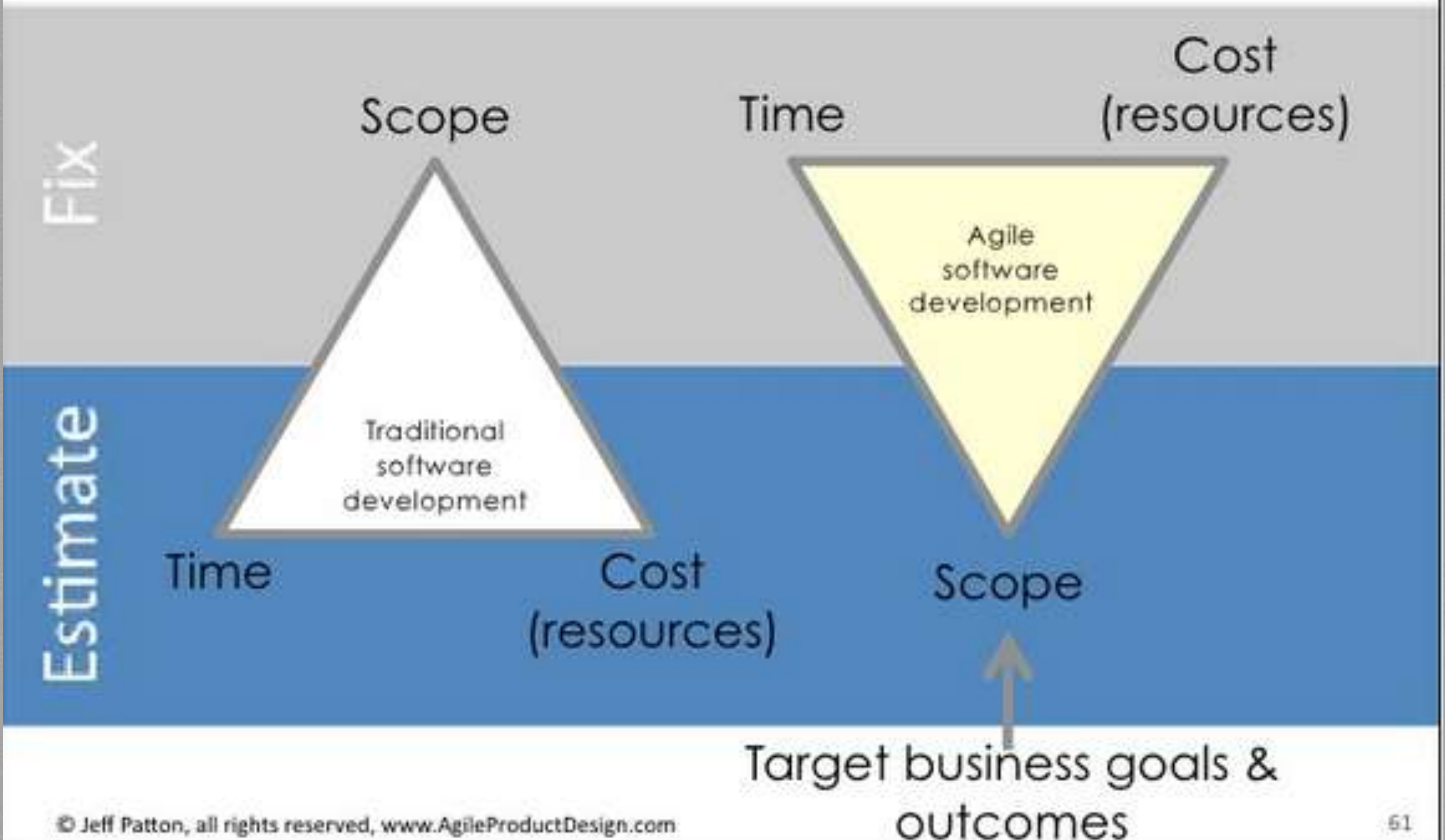
2

3

4

The agile release planning
Strategy\Backlog
Grooming
Regression testing
Testing strategy in each phase
The tester activities & team structure

Leverage a shared understanding of desired product goals to minimize scope while maximizing value



“incrementing” builds a bit at a time

incrementing calls for a fully formed idea.

And, doing it on time requires dead accurate estimation.

"iterating" builds a rough version, validates it, then slowly builds up quality

A more iterative allows you to move from vague idea to realization making course corrections as you go.

Kano explains three general classifications for product features: **must-haves**, **one-dimensionals**, and **delighters**

"This car has many flaws. Buy it anyway. It's so much fun to drive"

-- from a NY Times review of the Mini Cooper

Must-haves

The products must have this features for me to be consider the product acceptable

One-dimensionals

The more of this I get, the better

Delighters

I love this element of the product!

Divide release design & development into three phases

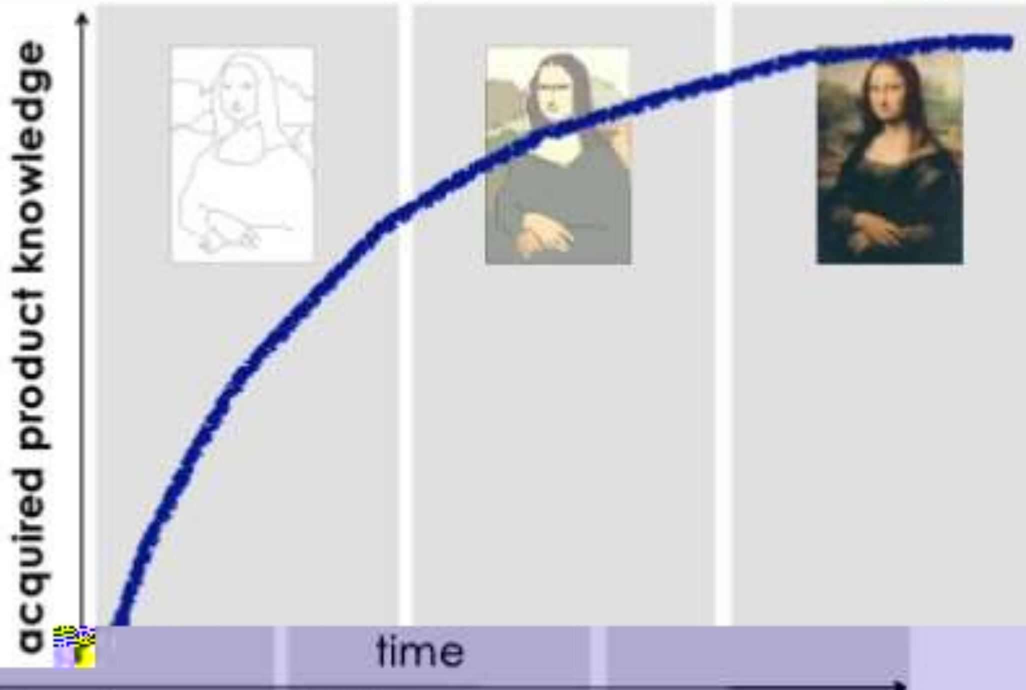
Opening Game: Build a simple system span of necessary features first – the walking skeleton

Mid-Game: Add flexibility and safety next

End Game: Finish with comfort, performance, and luxury

Reserve time in the remaining third for unforeseen additions and adaptations

Focusing on product knowledge acquired, We see the discovery benefit gained



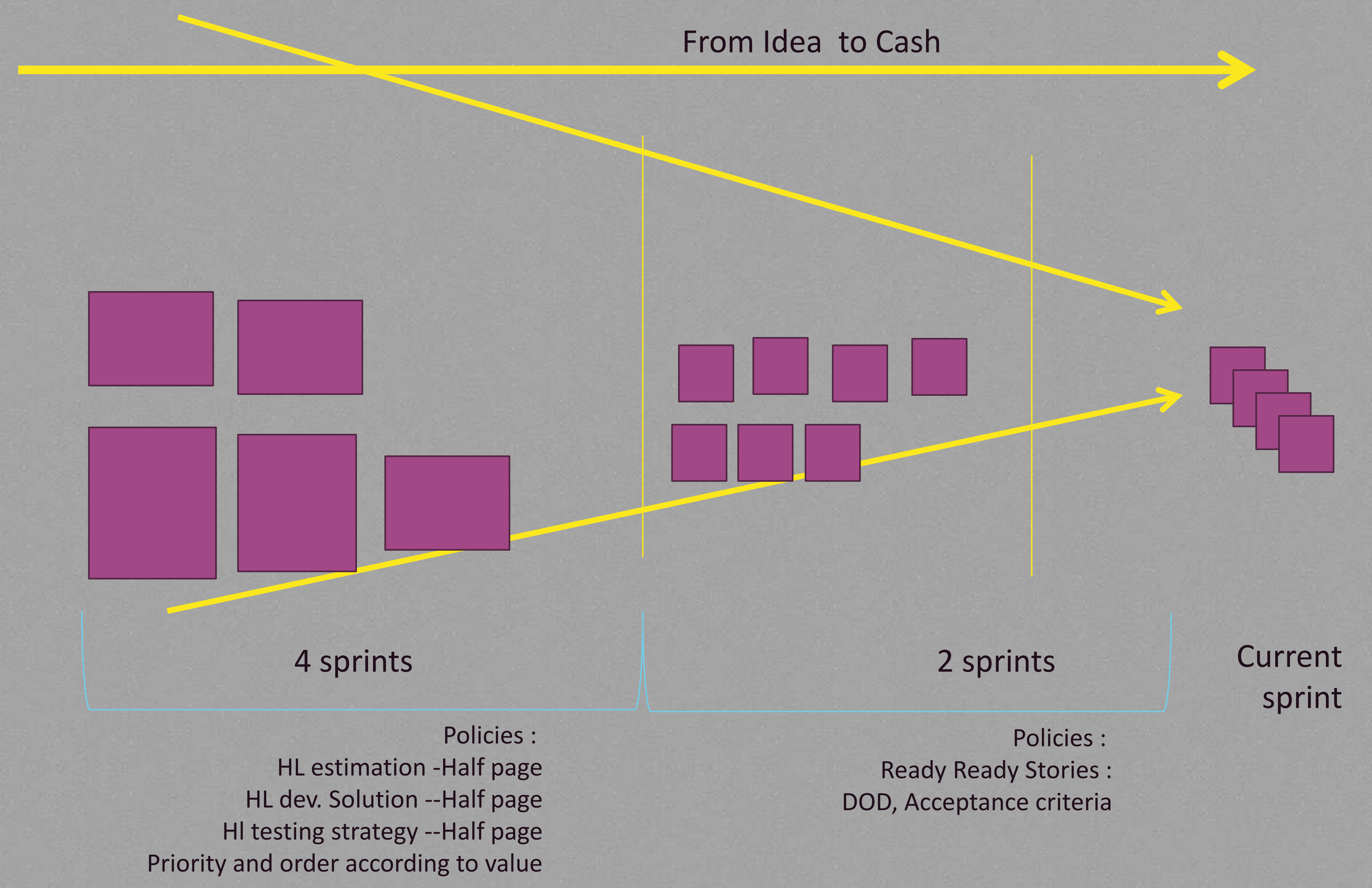
Opening Game
Early stories emphasize iteration and learning. We need to be sure we're building the right product

Mid Game
Once we're confident we have the "shape" of the product right, we begin to pile in value

End Game
Over time the value of stories begin to diminish signaling it's time for release

The inverse of risk is knowledge

Learning earlier if we're building the right product mitigates risk


From Idea to Cash
4 sprints
2 sprints
Current sprint
Policies :
HL estimation -Half page
HL dev. Solution --Half page
Hl testing strategy --Half page
Priority and order according to value
Policies :
Ready Ready Stories :
DOD, Acceptance criteria

The Hardening Sprint

the process of securing a system, adding a level of polishing and testing to the product/project that doesn't normally occur every sprint

Test Types example

	Type	Frequent	Measurements
Sanity	Integration	Daily	Before delivery of a build for testing to have 100% passed sanity scenarios
	End 2 End	Weekly	
Regression	Integration	Daily	Measure Success rate.
	End 2 End	Weekly	
	Manual	Weekly	
Progression	Service Level	Every Build	Measure Success rate.
	Integration	Daily	
	End 2 End	Weekly	
	Manual	Weekly	
Load	Integrated system	Daily	List of end to end flows and covered scenarios results
Stress	Integrated system	Weekly	

Automatic Test Types example

	Test Subject	Tool	Who	Frequent	Measurements
Unit Test	Every piece of code		Development	Every Build	Per iteration - graph of how many UT were added. Success rate per day. Code coverage.
Service Level Test	Business Components	Fitnesse	Development	Every Build	Definition of done for a user story is to have SLT. Success rate per day.
Integration Test	Integrated system| (Server, DB etc). Testing tools acts as a client	Fitnesse	QA	Daily	For each functional requirement have E2E end flows that covers it. Measure Success rate.
End 2 End	Integrated system against live client	QTP / Manual	QA	Daily / Weekly	List of end to end flows and covered scenarios results Measure Success rate.
PerformanceTe sting	Integrated system (Server, DB etc). Testing tools acts as a client	Load Runner	QA	Daily / Weekly	List of end to end flows and covered scenarios results