Certified Professional
Basic Agile Testing

CP-BAT

Agile Testing Alliance
version 3.2 June 2018

CERTIFICATION

Agile Testing Alliance Certifications

Agile Certification
- CP-BAT
- CP-MAT
- CP-AAT

Continuous Testing Certification
- CP-CCT

Tools Certification
- CP-SAT
- CP-SAT ADVANCED
- CP-AMT

Mobile Certification
- CP-M-AAT
- CP-M-MAT

Security Certification
- CP-AAST

Test Data Certification
- CP-TDM

Web Services Certification
- CP-WST

Knowledge with experience is power;
Certification is just a by-product.

Knowledge, delivery and certifications are consciously designed to focus on **“PRACTICAL AGILE TESTING”** for **ALL roles in agile**.

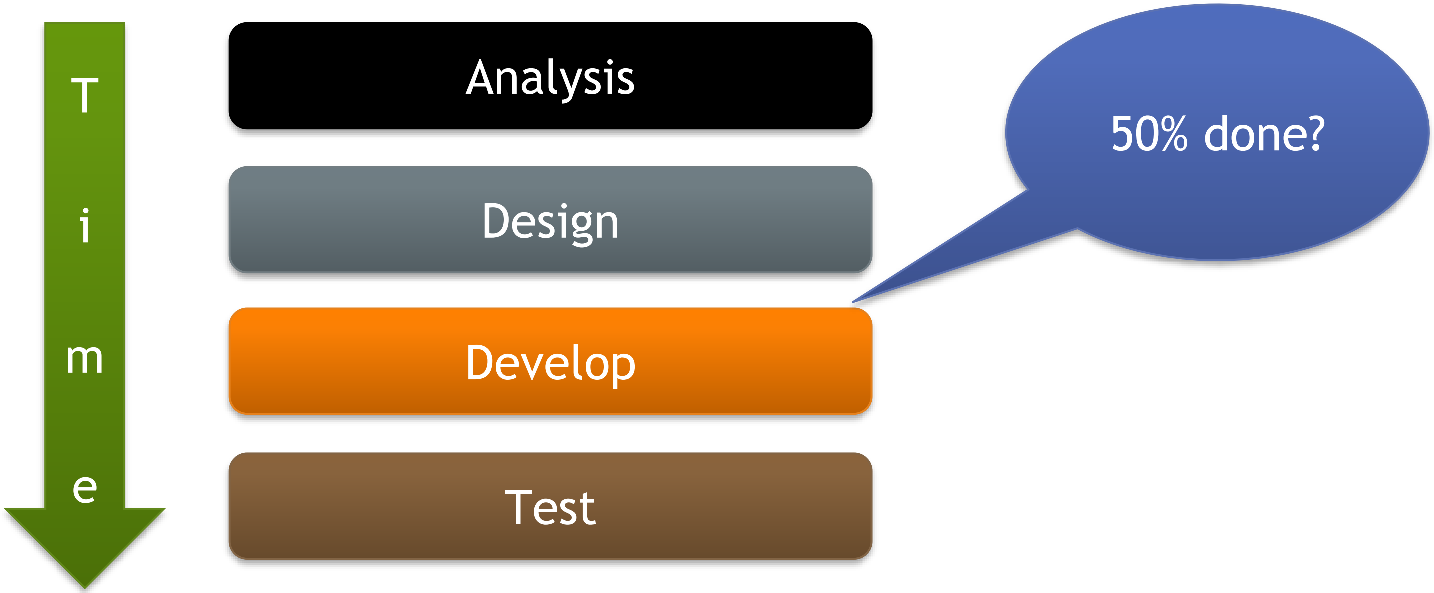
CP-BAT

- Certification Criteria
  - Quantitative Assessment (Total 40 Marks)
    - 40 questions / 60 min, 1 mark each, no negative marking
  - Passing Criterian 60%

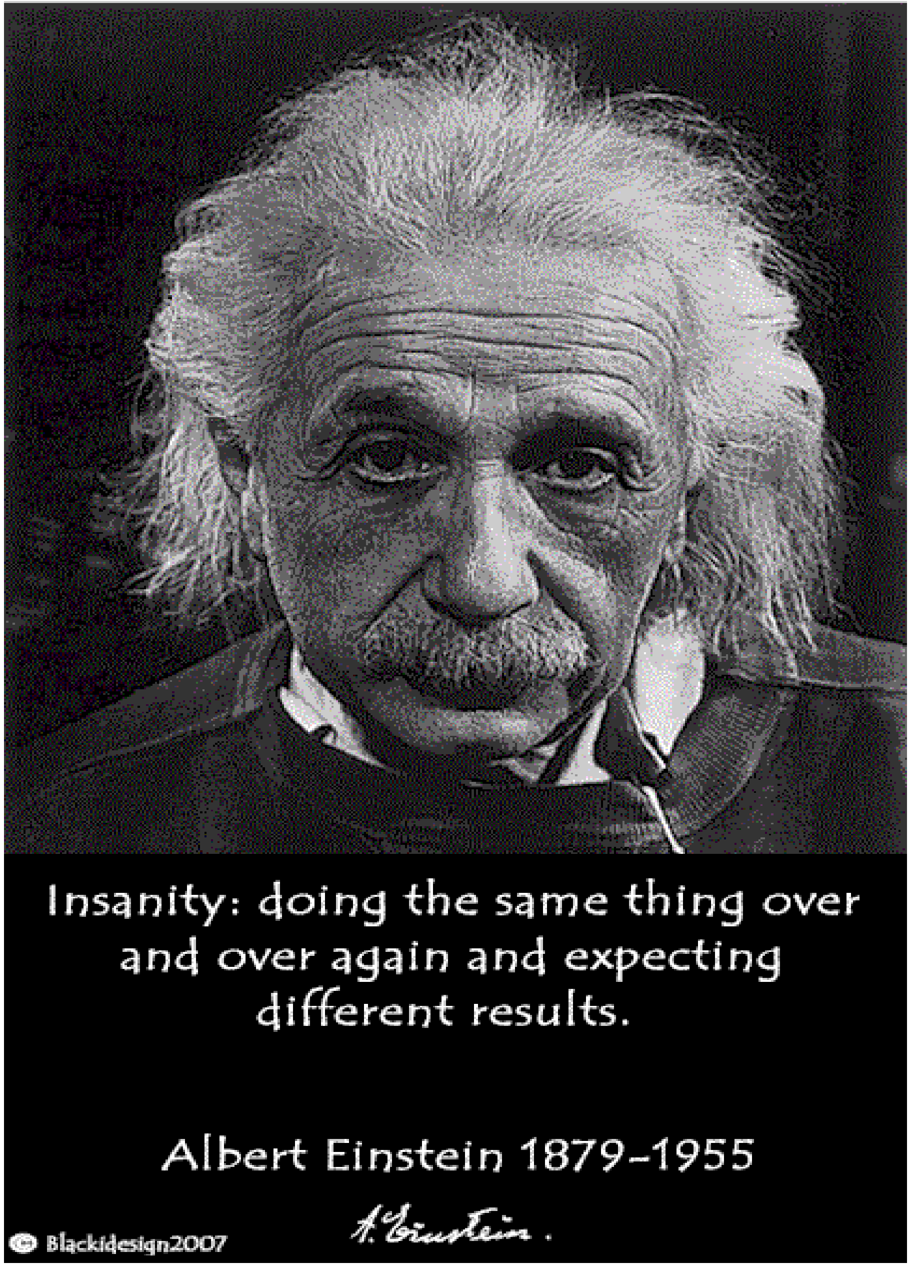
AGILE FUNDAMENTALS

AGILE HISTORY, MANIFESTO & PRINCIPLES

Waterfall



1970, Dr. Winston Royce published “Managing the Development of Large Software Systems”, where the waterfall is first documented! he said “I believe in this concept, but the implementation described above is **risky and invites failure**”



Insanity: doing the same thing over and over again and expecting different results.

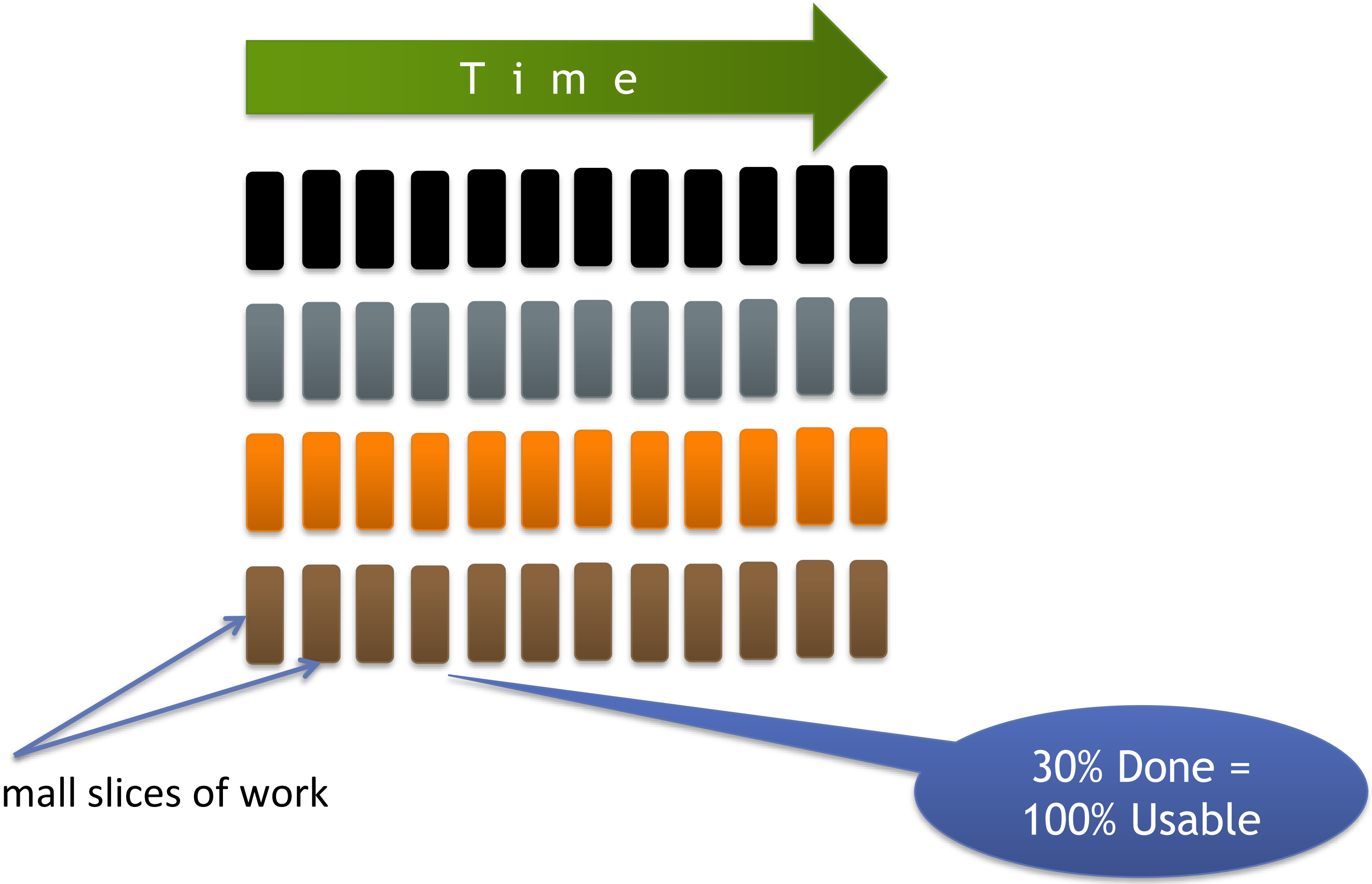
Albert Einstein 1879-1955

A. Einstein.

© Blackidesign2007

http://in-the-flow.com/wp-content/uploads/2012/01/insanity.gif

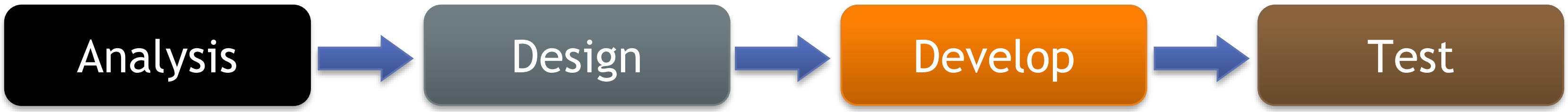
Agile



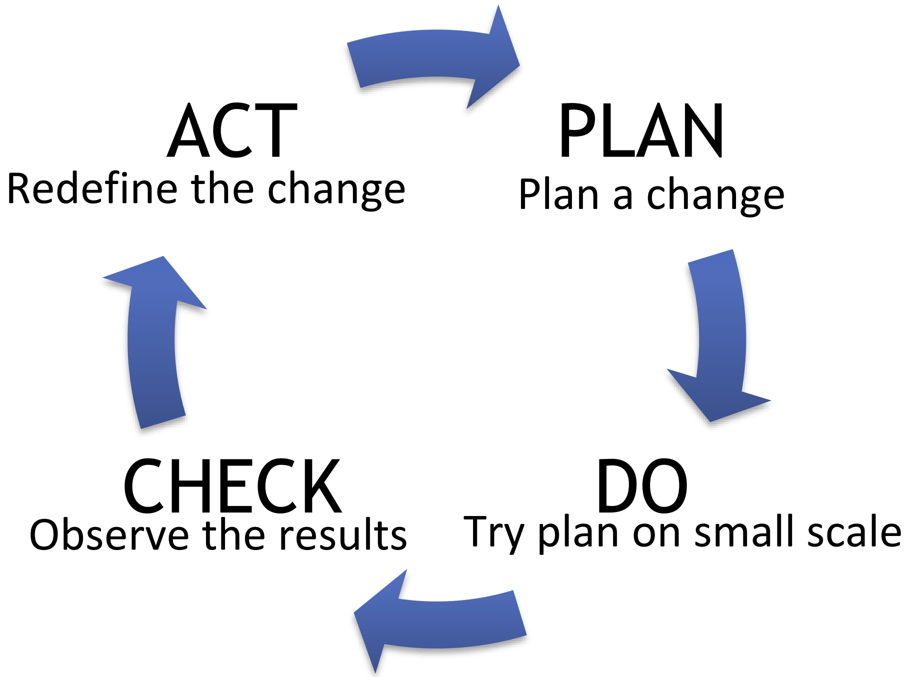
End to end small slices of work

30% Done =
100% Usable

Defined VS Empirical



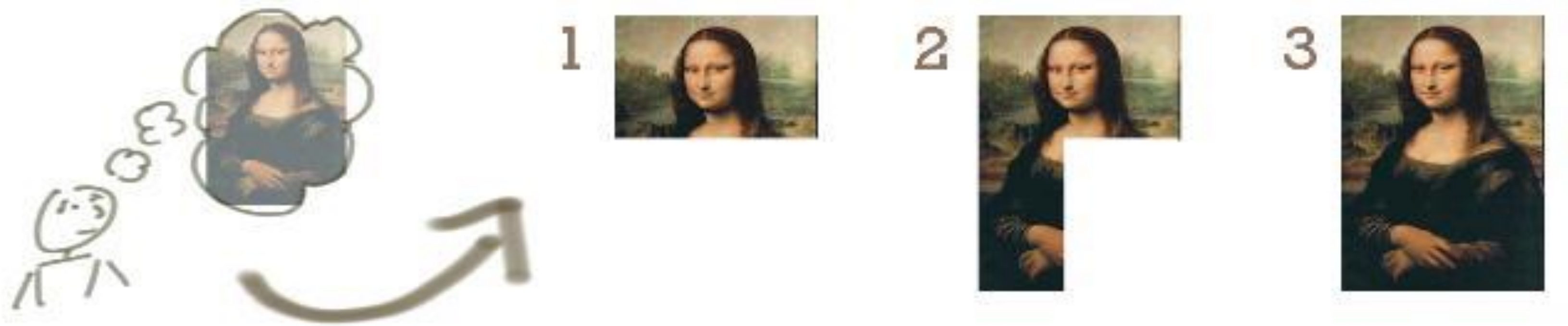
PDCA is also called as Deming Cycle



Empirical means guided by experience or experiment.

(Inspect and Adapt)

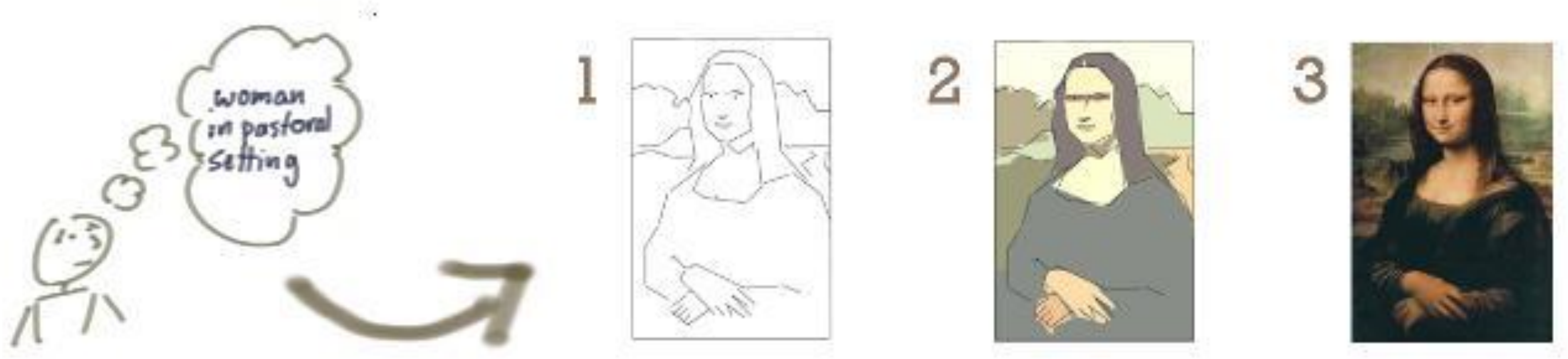
Incremental



Credits: Jeff Patton

- ✓ Build a system gradually
- ✓ Demonstrate progress

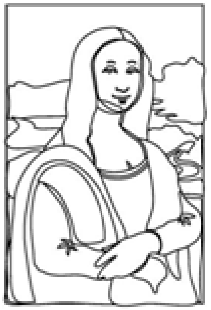
Iterative



Credits: Jeff Patton

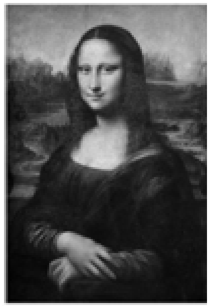
- ✓ Multiple releases (every month or so)
- ✓ Iterations (usually one or two weeks)

Incrementing + Iterating



Leonardo sketches what he intends to do and goes to the patron, asking, “How`s this going to work for you?” The patron says, “No, no, no. She can`t be looking right, she has to be looking left!” Fortunately, Leonardo has not done too much work yet, so this is easy to change.

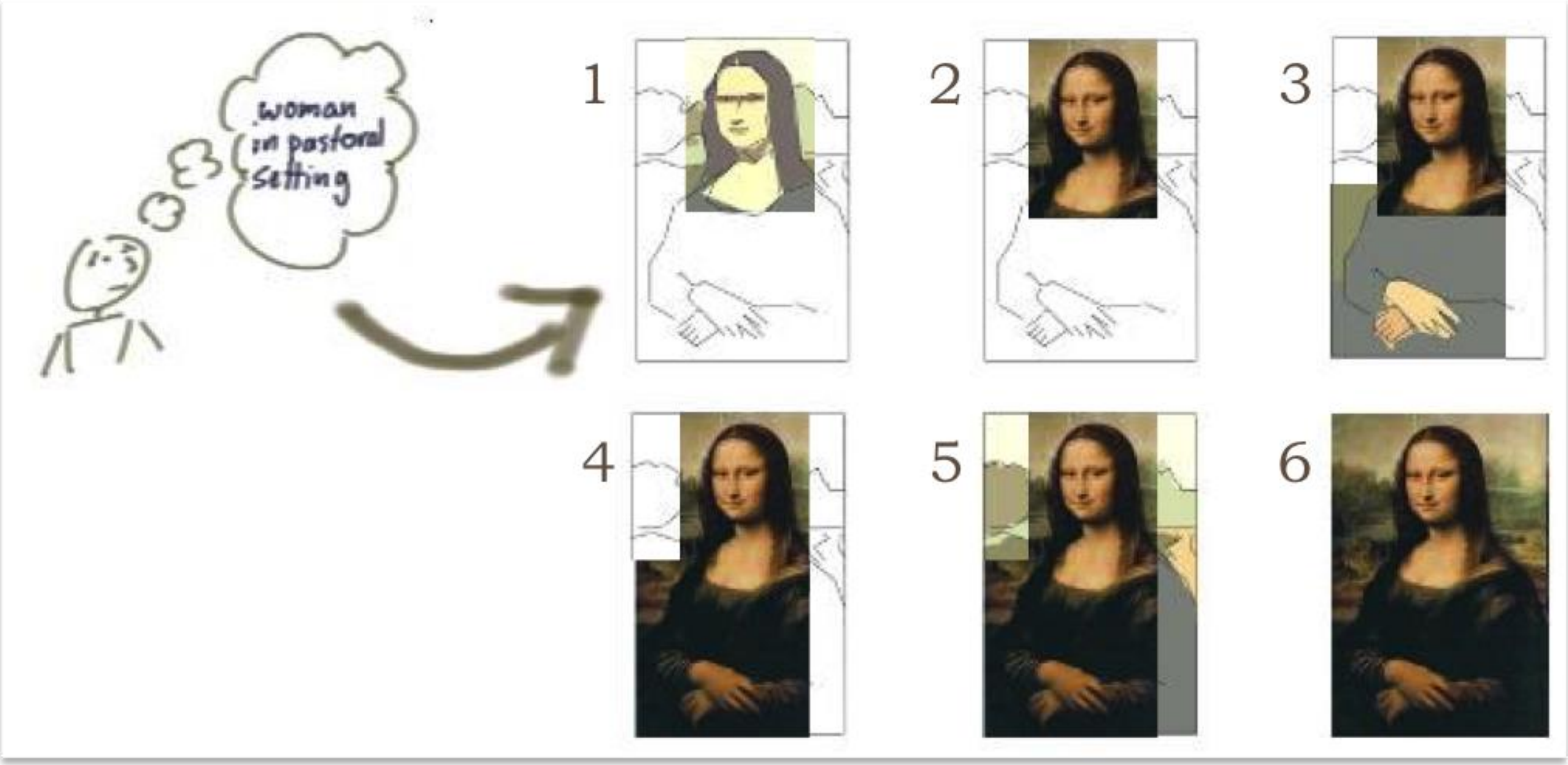
Leonardo goes away, reverses the picture and does some color and detail (Figure 7). He goes back to the patron: “By cost, I`m about one-third done. What do you think now?” The patron says, “No, you can`t make her head look that big! Make it look more balanced with her body size.”



Leonardo goes away and finishes the painting and delivers the final product.

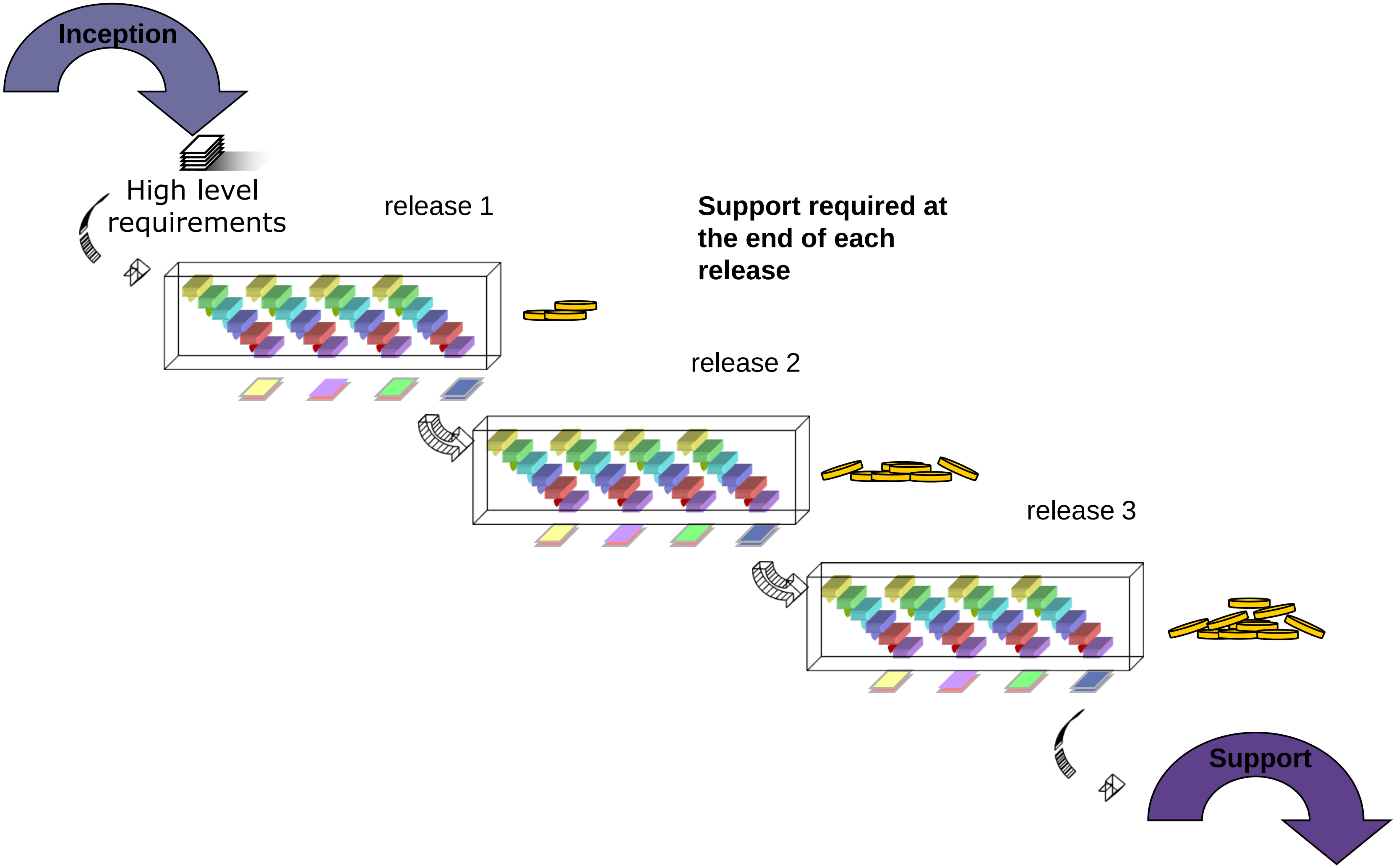
http://www.stsc.hill.af.mil/crosstalk/2010/05/0805Cockburn.html

Incrementing and Iterating



Credits to Jeff Patton
Explained in : http://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa

Iterative and Incremental Model



Support required at the end of each release

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

INDIVIDUALS AND INTERACTIONS —OVER PROCESSES AND TOOLS
WORKING SOFTWARE —OVER COMPREHENSIVE DOCUMENTATION
CUSTOMER COLLABORATION — OVER CONTRACT NEGOTIATION
RESPONDING TO CHANGE —OVER FOLLOWING A PLAN

That is, while there is value in the items on the right, we value the items on the left more

Credit: ThoughtWorks

Agile Principles

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

2. Welcome changing requirements, even late in the 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 preference to the short time scale

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 agiltiy

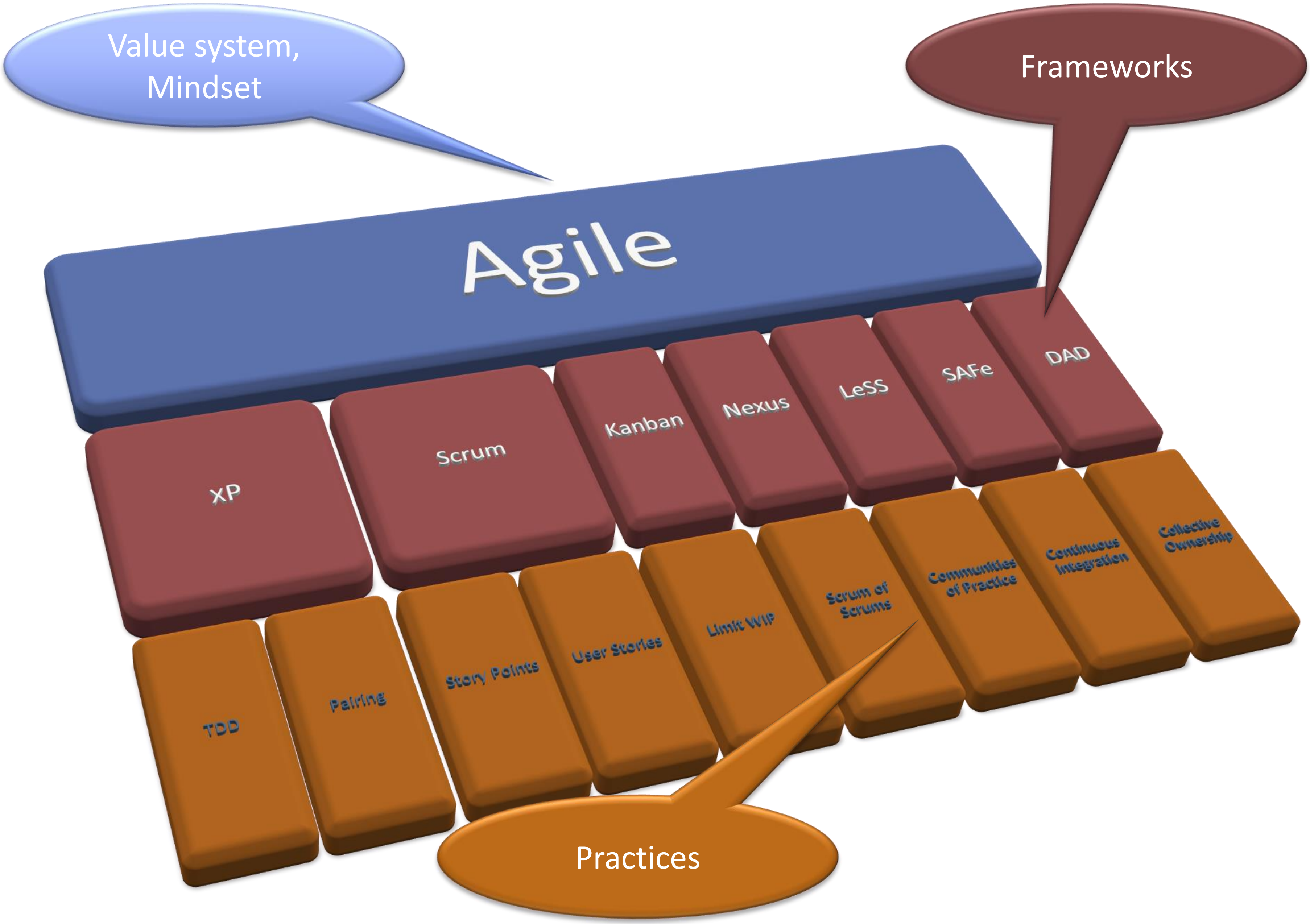
10. Simplicity – the art of maximising the amount of work not done – is essential

11. The best architecture, requirements and designs emerge from self-organising teams

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

Agile Is Mindset

LEARN DIFFERENT AGILE MODELS



Value system, Mindset

Frameworks

Agile

XP

Scrum

Kanban

Nexus

LeSS

SAFe

DAD

TDD

Pairing

Story Points

User Stories

Limit WIP

Scrum of Scrums

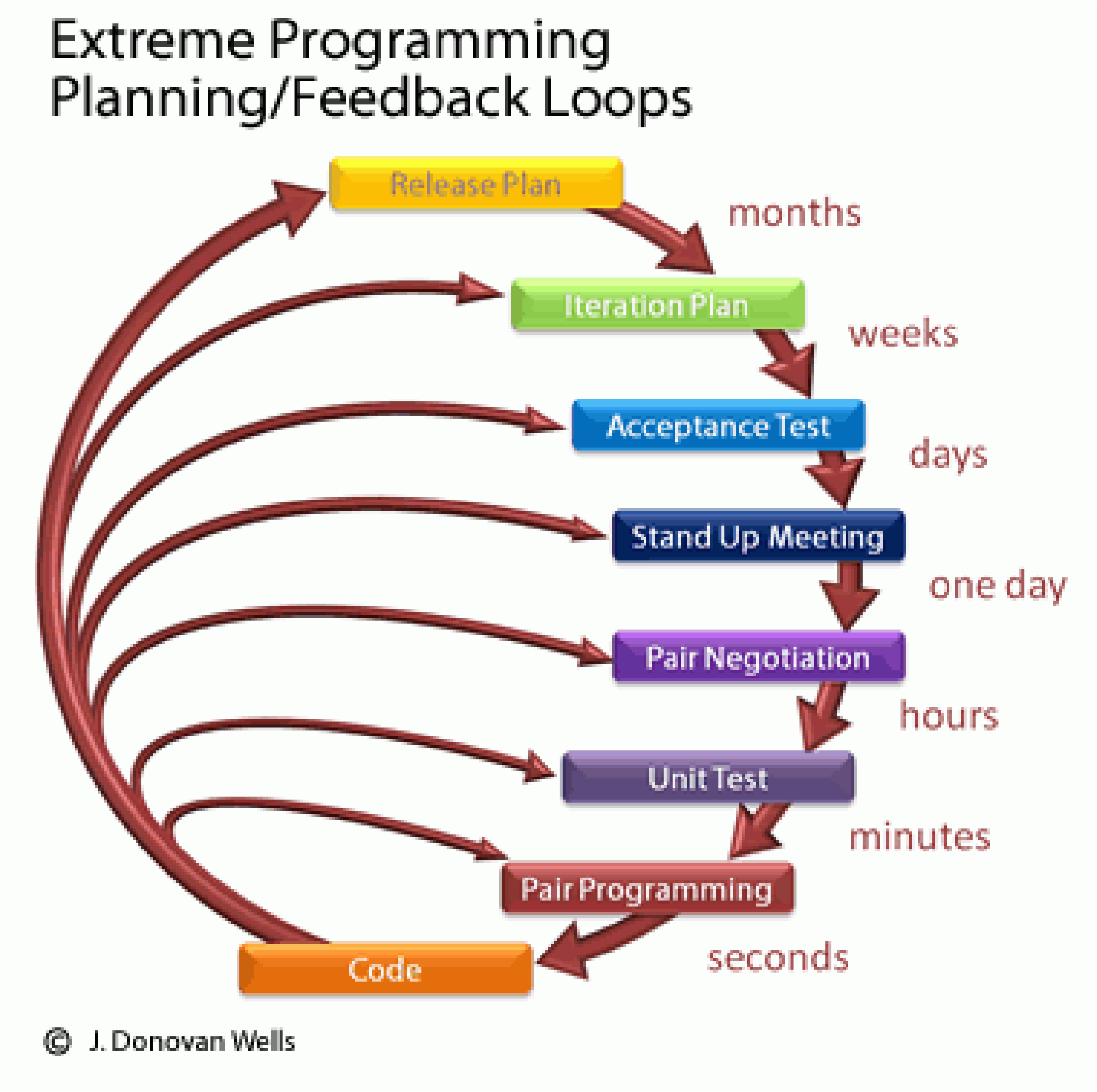
Communities of Practice

Continuous Integration

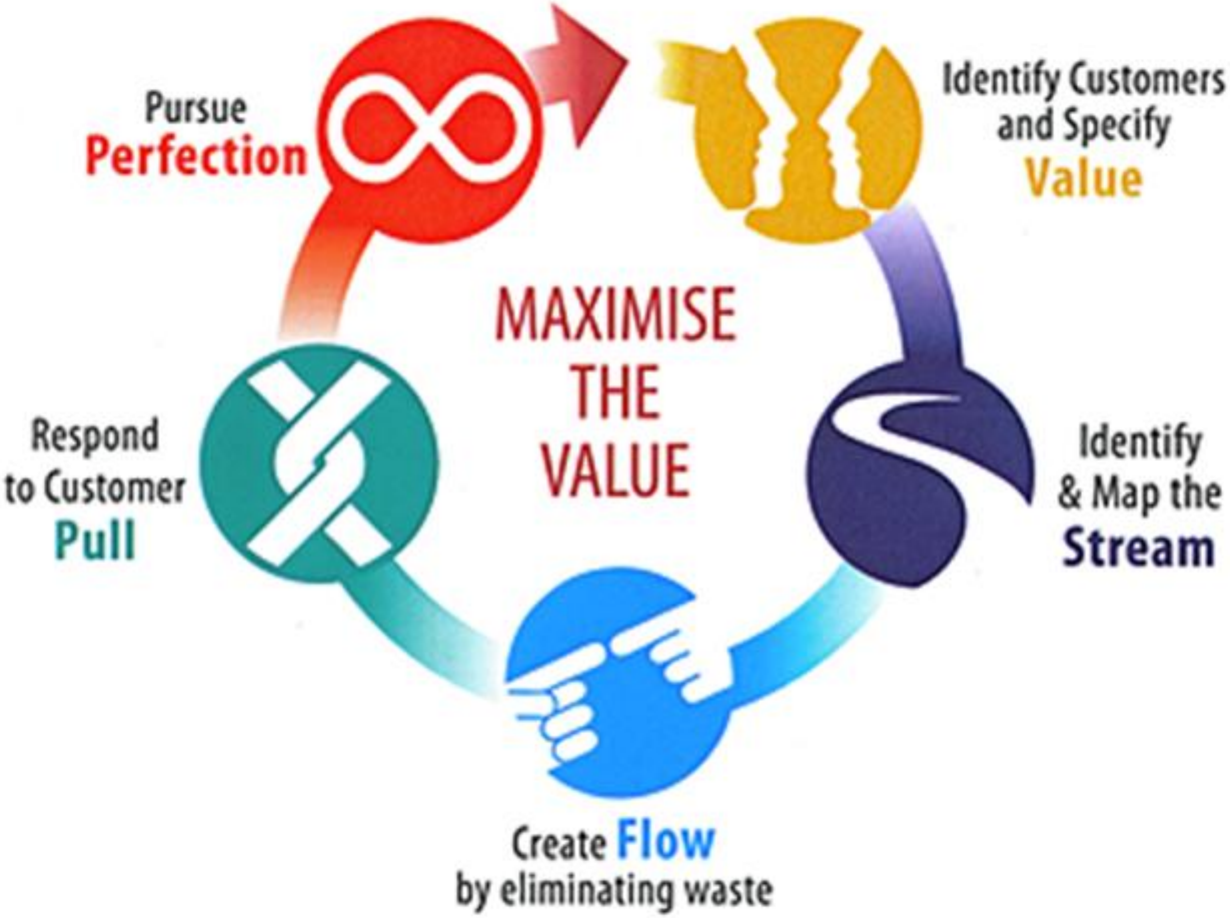
Collective Ownership

Practices

XP


Extreme Programming
Planning/Feedback Loops
Release Plan
months
Iteration Plan
weeks
Acceptance Test
days
Stand Up Meeting
one day
Pair Negotiation
hours
Unit Test
minutes
Pair Programming
seconds
Code
© J. Donovan Wells

Lean


Pursue Perfection
Identify Customers and Specify Value
MAXIMISE THE VALUE
Respond to Customer Pull
Identify & Map the Stream
Create Flow by eliminating waste

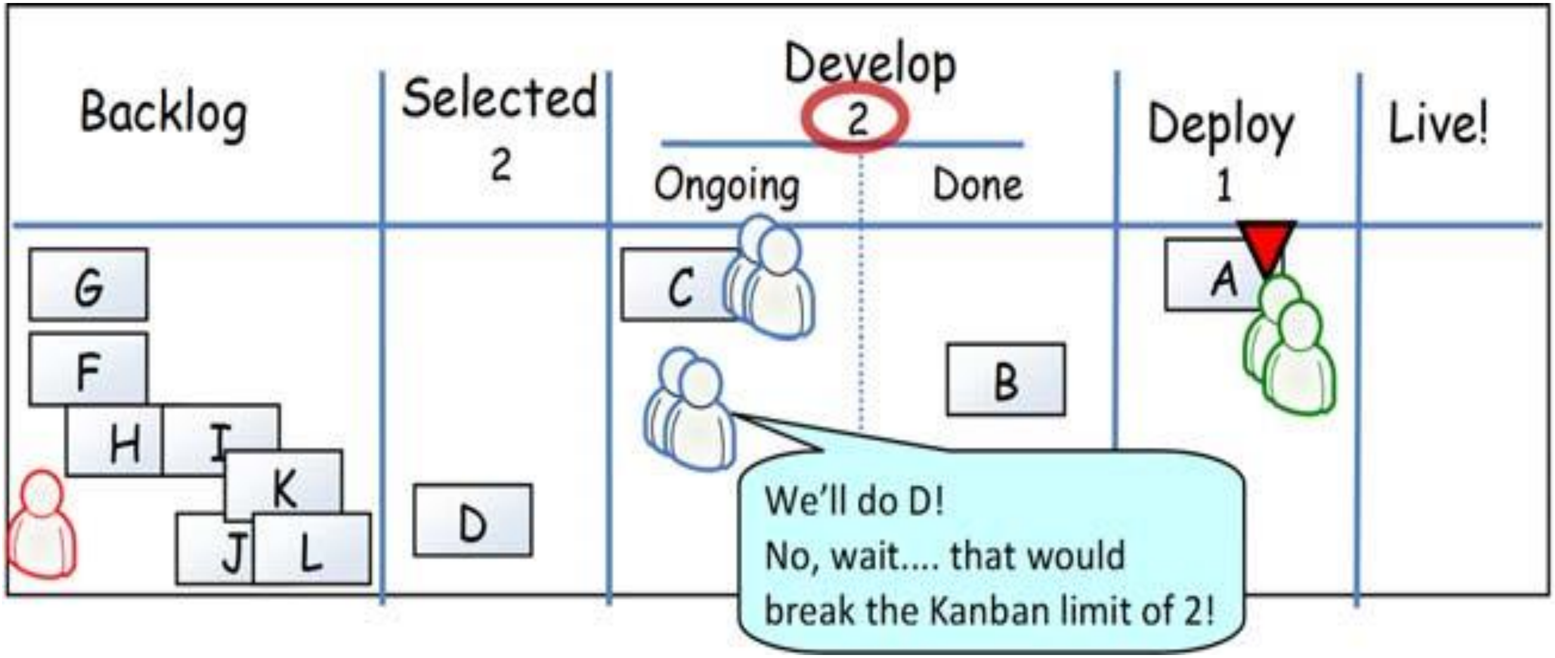
http://www.cardiff.ac.uk/lean/images/image-186652-web.jpg

Kanban

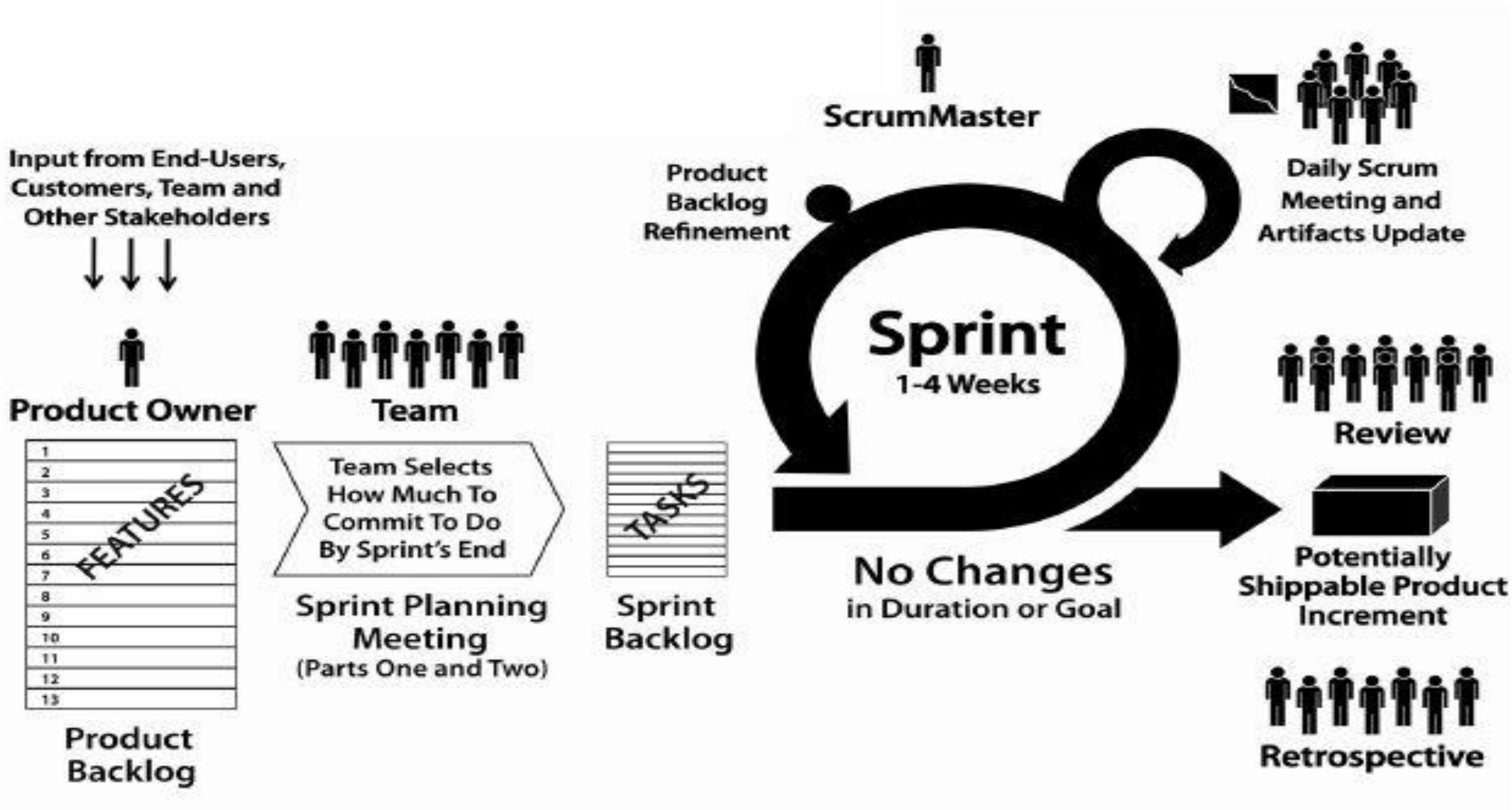
Visualize workflow

Limit work in progress (WIP)

Measure the lead time



SCRUM


Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
1
2
3
4
5
6
7
8
9
10
11
12
13
FEATURES
Product Backlog
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting
(Parts One and Two)
TASKS
Sprint Backlog
Product Backlog Refinement
ScrumMaster
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

http://javamaster.files.wordpress.com/2009/07/scrum1.png

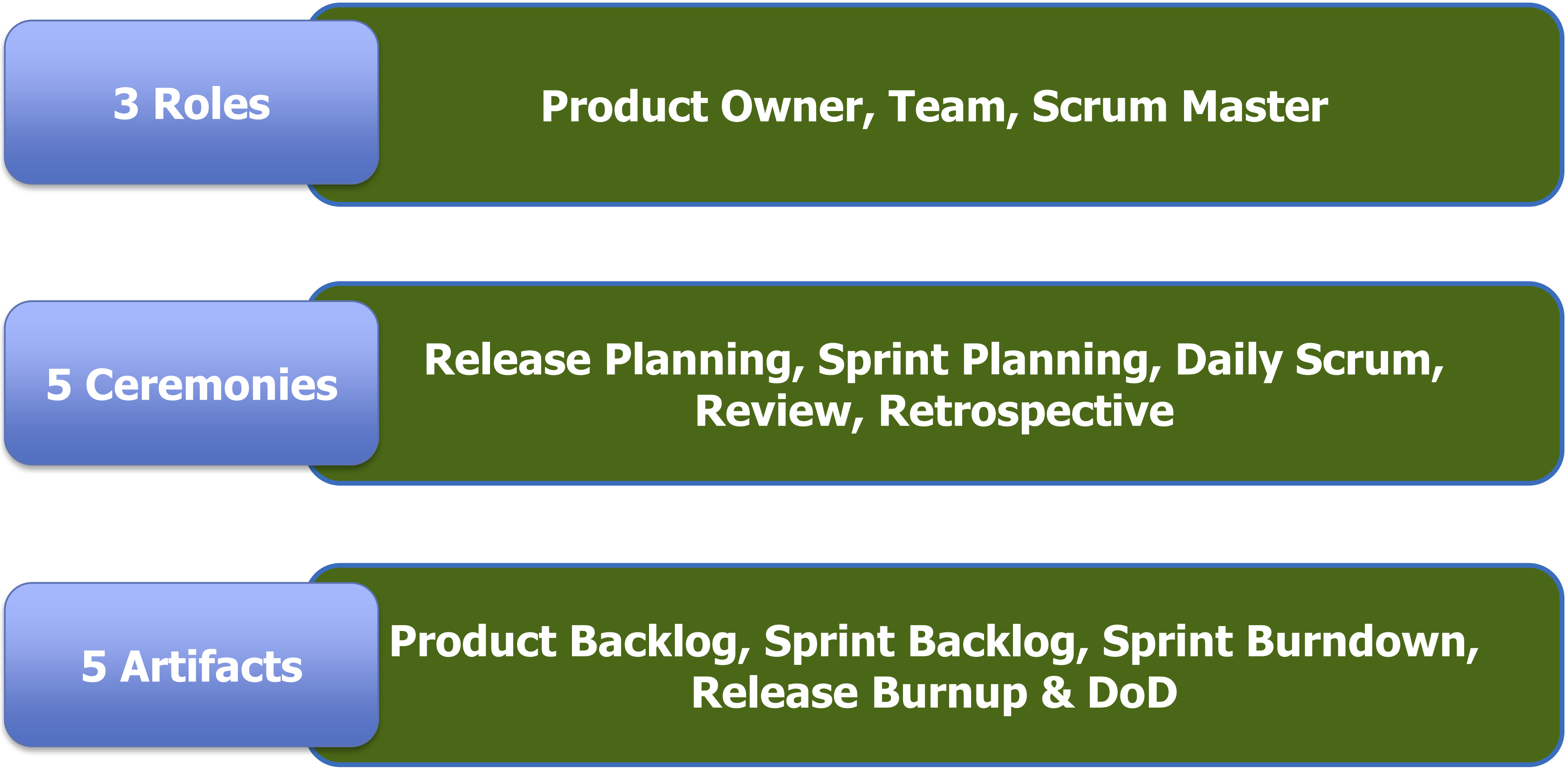
Scrum – 3 5 5

3 Roles

5 Ceremonies

5 Artifacts

Scrum: 3 Roles, 5 Ceremonies, 5 Artifacts



3 Roles

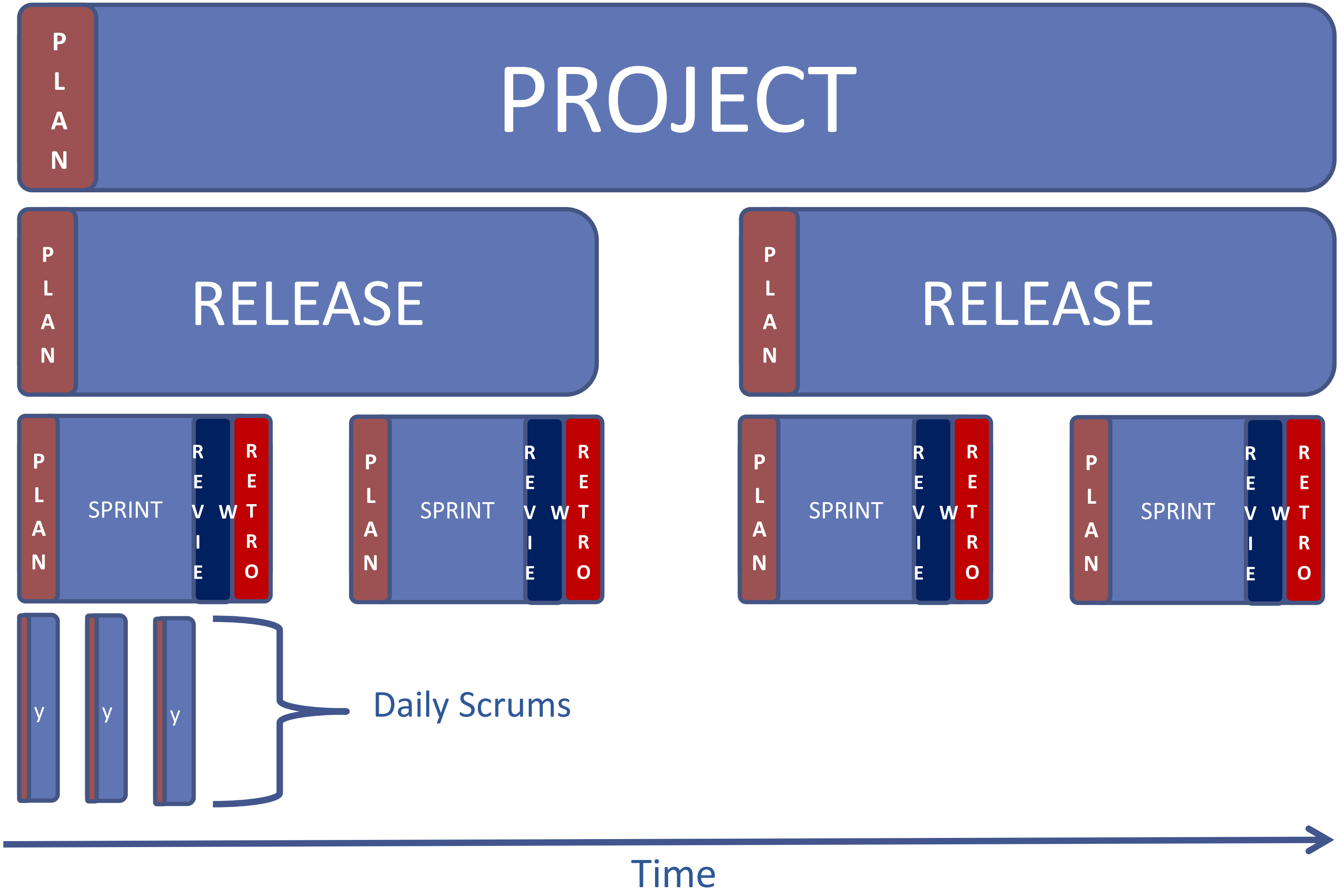
Product Owner, Team, Scrum Master

5 Ceremonies

Release Planning, Sprint Planning, Daily Scrum, Review, Retrospective

5 Artifacts

Product Backlog, Sprint Backlog, Sprint Burndown, Release Burnup & DoD



PLAN

PROJECT

PLAN

RELEASE

PLAN

RELEASE

PLAN

SPRINT

REVIEW

RETRO

PLAN

SPRINT

REVIEW

RETRO

PLAN

SPRINT

REVIEW

RETRO

PLAN

SPRINT

REVIEW

RETRO

y

y

y

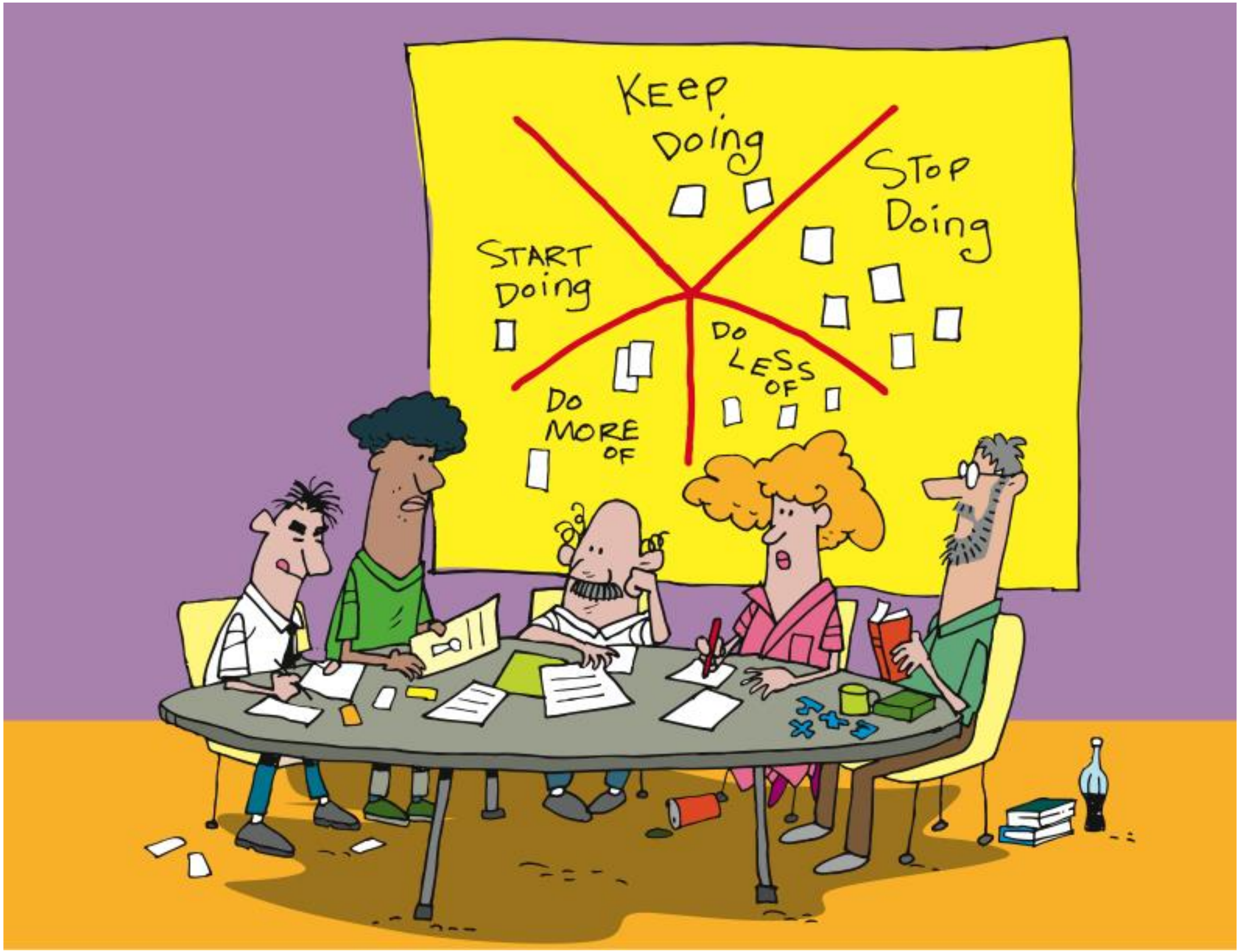
Daily Scrums

Time

Daily Scrum

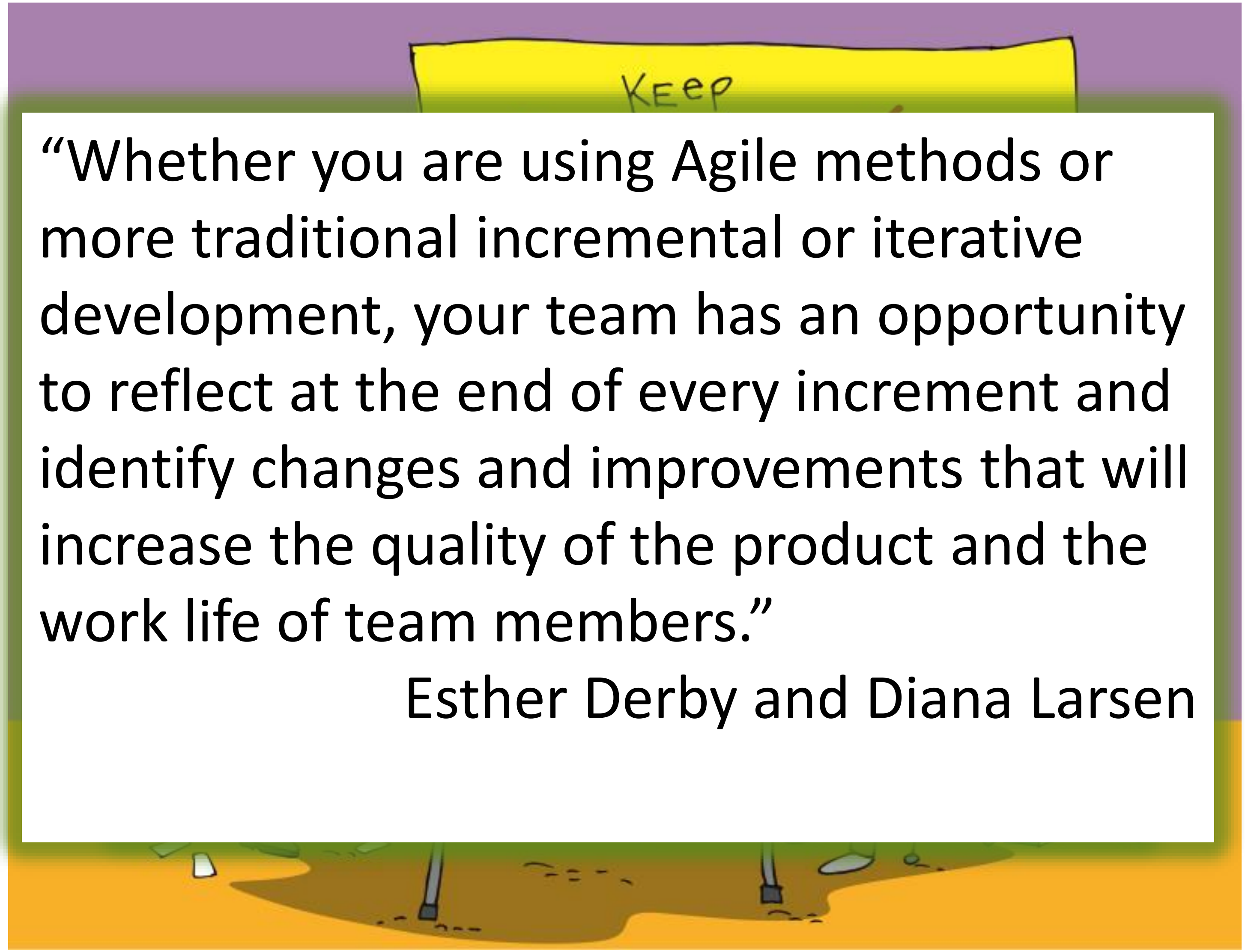
- Why?
  - Zoom out and sync up
  - Are we on track?
  - I need help
  - Can I help someone?
- How?
  - 15 min stand up
  - 3 Questions
  - Walk the wall
  - Talking token
  - Time out siren / sign
  - Wake up sign / soft toy
- Who?
  - Team

Retrospectives



http://www.thekua.com/rant/wp-content/uploads/2006/03/StarTechnique.gif

Retrospectives



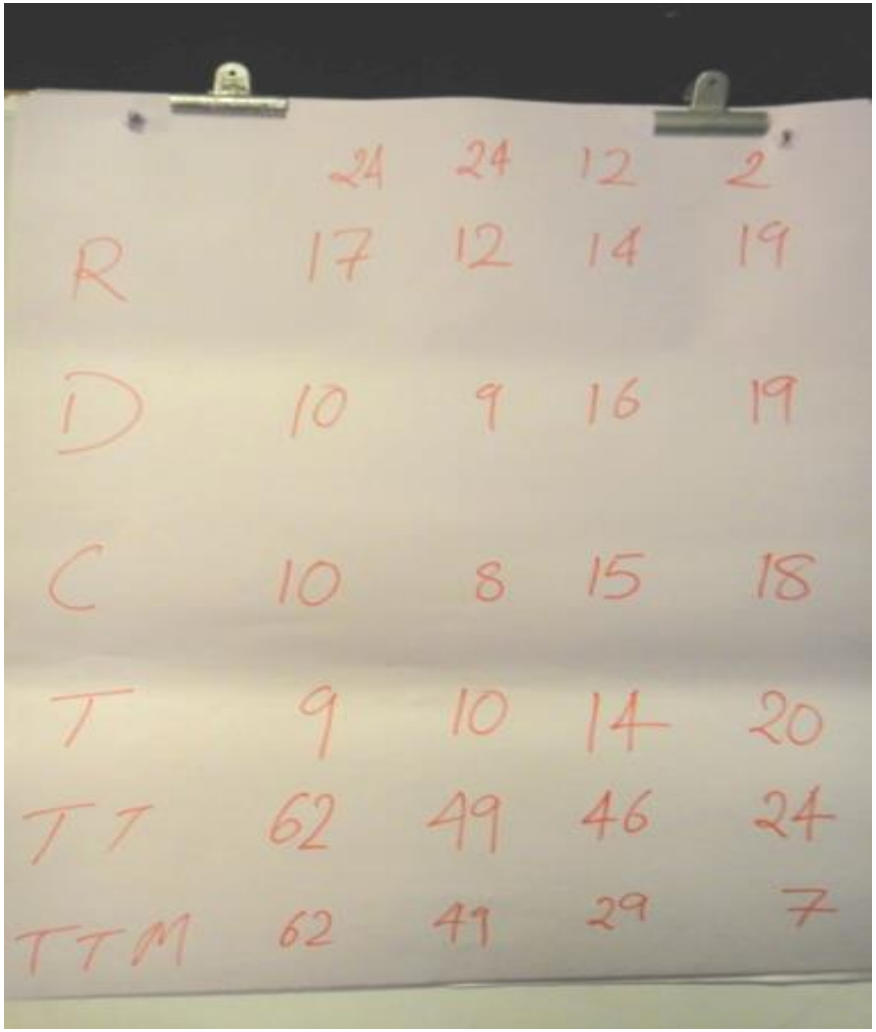
"Whether you are using Agile methods or more traditional incremental or iterative development, your team has an opportunity to reflect at the end of every increment and identify changes and improvements that will increase the quality of the product and the work life of team members."

Esther Derby and Diana Larsen

http://www.thekua.com/rant/wp-content/uploads/2006/03/StarTechnique.gif

EXPERIENCE AGILITY

Penny Game Chart



	24	24	12	2
R	17	12	14	19
D	10	9	16	19
C	10	8	15	18
T	9	10	14	20
TT	62	49	46	24
TTM	62	41	29	7

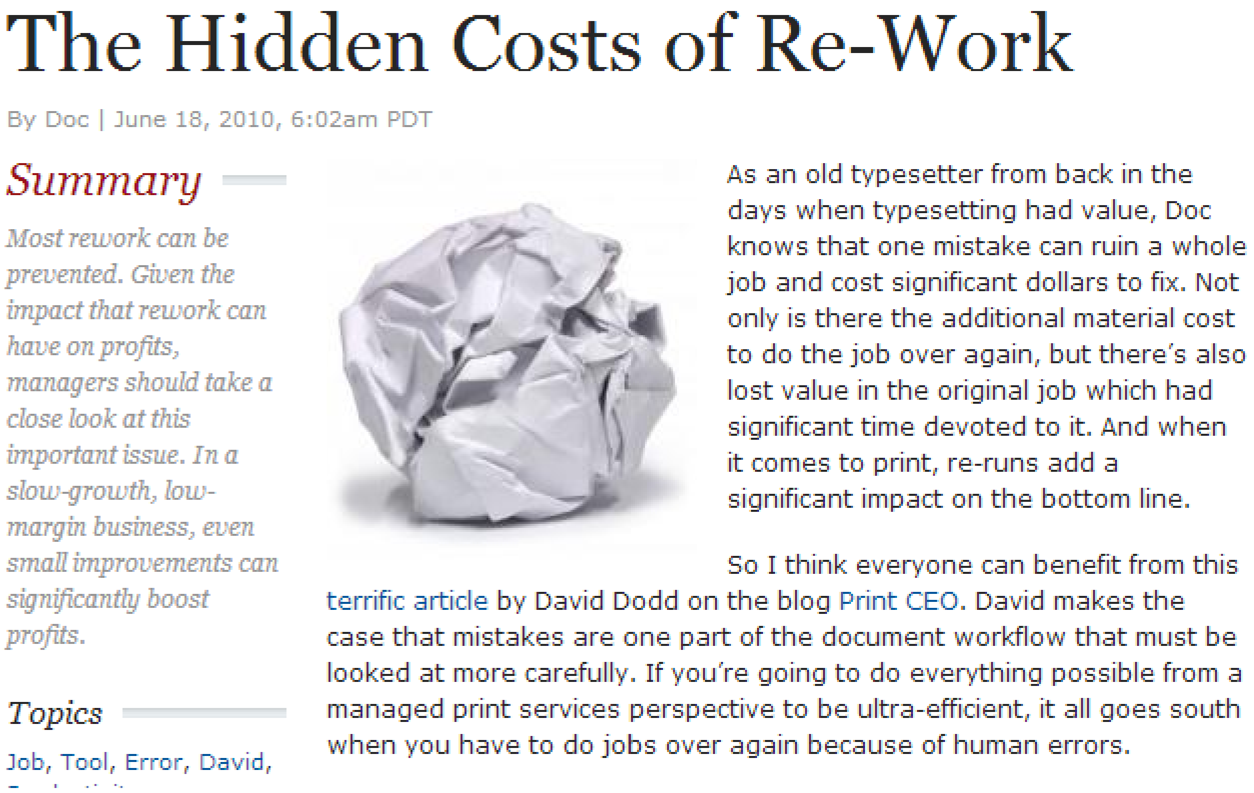
What Did We Learn?

- Wait is the biggest waste
- Agile helps eliminate waste
- Agile makes everyone working all the time
- Parallelism is the key
- Smaller the batch size lesser the waste
- Timebox helps limit batch size
- Agile = X % done, 100% usable
- Time to market is reduced drastically
- Total time of the project has reduced to less than half

TESTING FUNDAMENTALS

Testing Fundamentals

- Need for Testing



The Hidden Costs of Re-Work

By Doc | June 18, 2010, 6:02am PDT

Summary

Most rework can be prevented. Given the impact that rework can have on profits, managers should take a close look at this important issue. In a slow-growth, low-margin business, even small improvements can significantly boost profits.

As an old typesetter from back in the days when typesetting had value, Doc knows that one mistake can ruin a whole job and cost significant dollars to fix. Not only is there the additional material cost to do the job over again, but there's also lost value in the original job which had significant time devoted to it. And when it comes to print, re-runs add a significant impact on the bottom line.

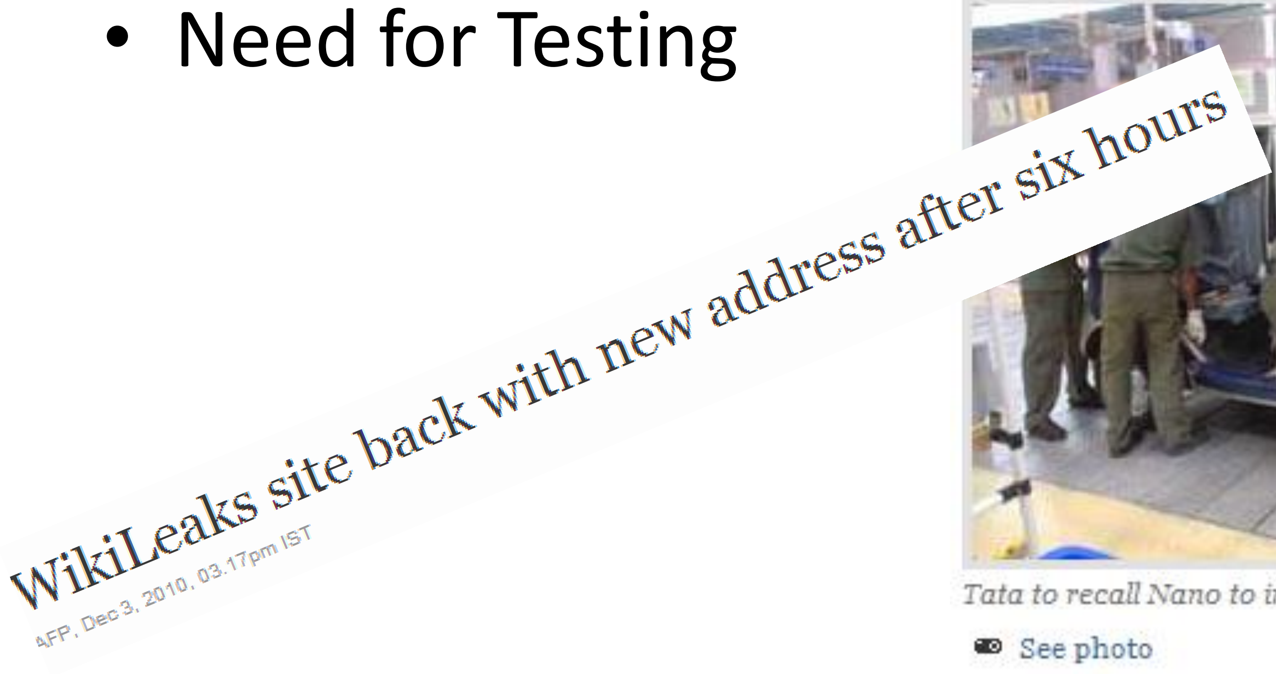
So I think everyone can benefit from this terrific article by David Dodd on the blog Print CEO. David makes the case that mistakes are one part of the document workflow that must be looked at more carefully. If you're going to do everything possible from a managed print services perspective to be ultra-efficient, it all goes south when you have to do jobs over again because of human errors.

Topics

Job, Tool, Error, David,

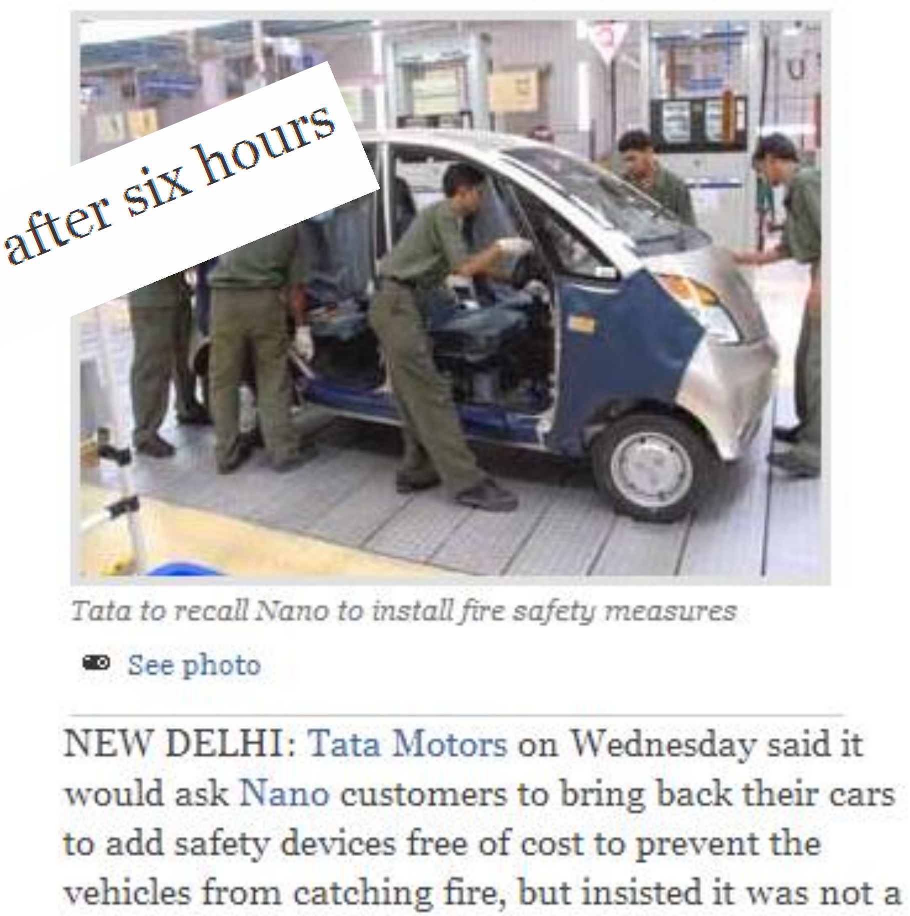
Testing Fundamentals

- Need for Testing



WikiLeaks site back with new address after six hours

AFP, Dec 3, 2010, 03.17pm IST



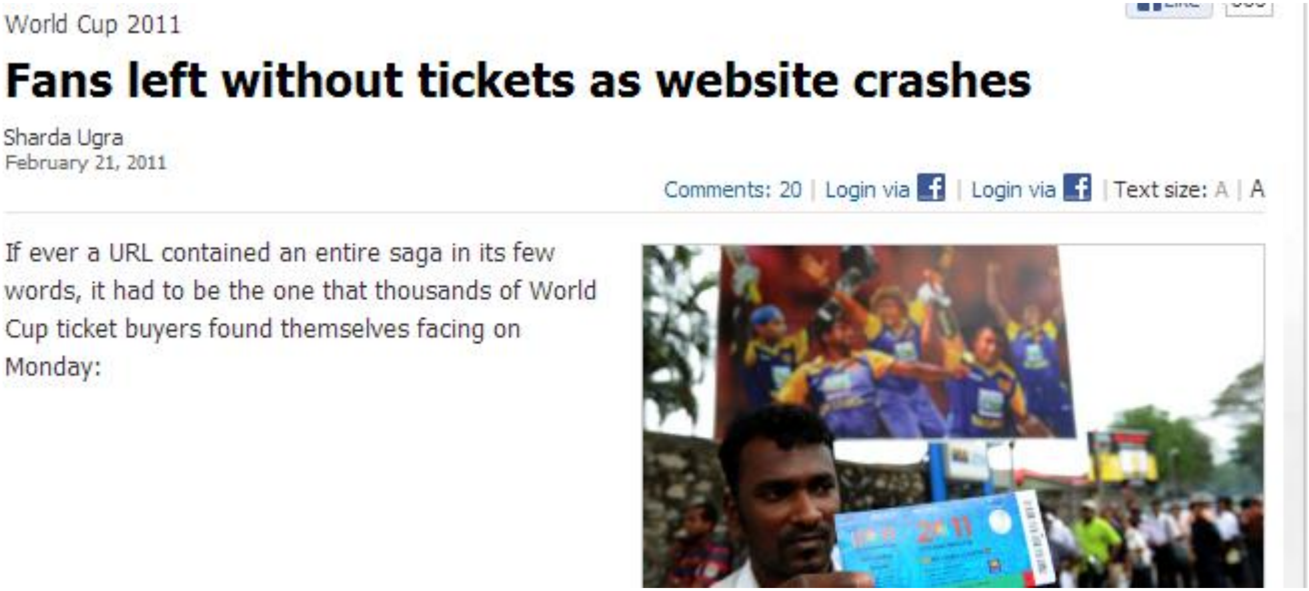
Tata to recall Nano to install fire safety measures

See photo

NEW DELHI: Tata Motors on Wednesday said it would ask Nano customers to bring back their cars to add safety devices free of cost to prevent the vehicles from catching fire, but insisted it was not a

Testing Fundamentals

- Need for Testing



World Cup 2011

Fans left without tickets as website crashes

Sharda Ugra
February 21, 2011

Comments: 20 | Login via f | Login via f | Text size: A | A

If ever a URL contained an entire saga in its few words, it had to be the one that thousands of World Cup ticket buyers found themselves facing on Monday:

Testing Fundamentals

Causes of software failures

- **Human error**
  A defect was introduced into the software code, the data or the configuration parameters

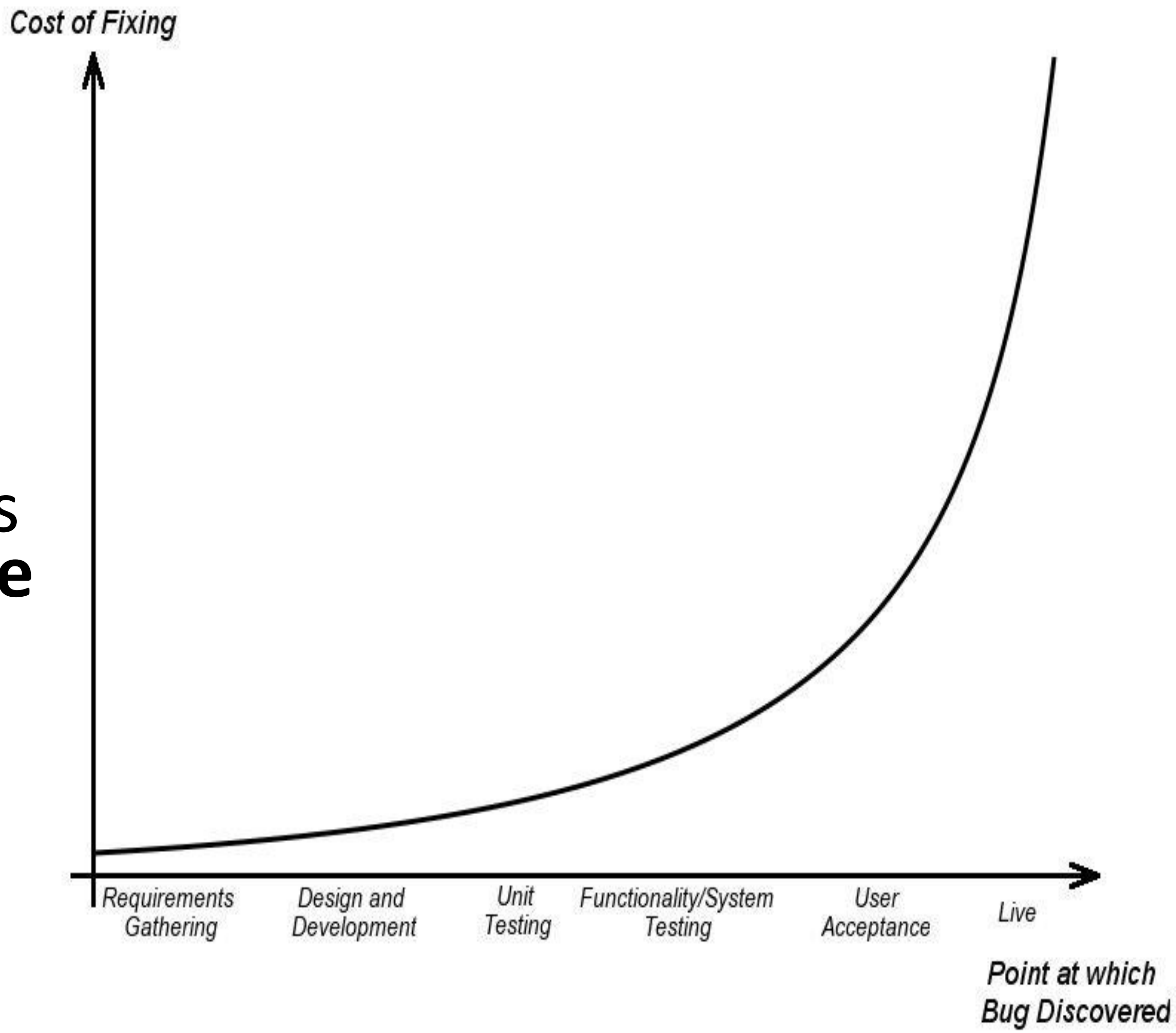
  Causes of human error
  time pressure, excessive demands because of complexity, distractions

- **Environmental conditions**
  changes of environmental conditions

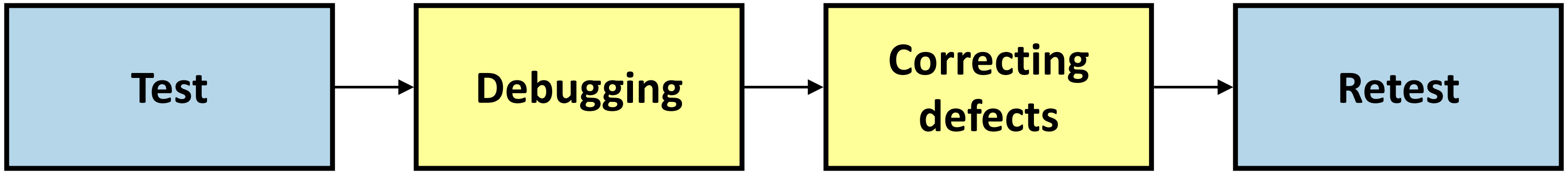
  Causes of negative environmental conditions
  radiation, magnetism, electronic fields and pollution
  sun spots, hard disk crashes, power fluctuations

Cost of Defects

- The **costs** of fixing defects **increase with** the **time** they remain in the system.
- Detecting errors at an **early stage** allows for error correction at reduced costs



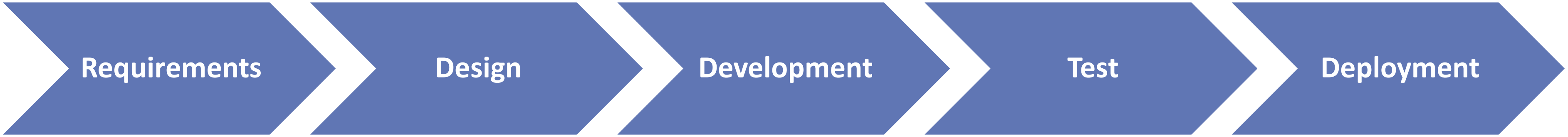
Testing and Debugging



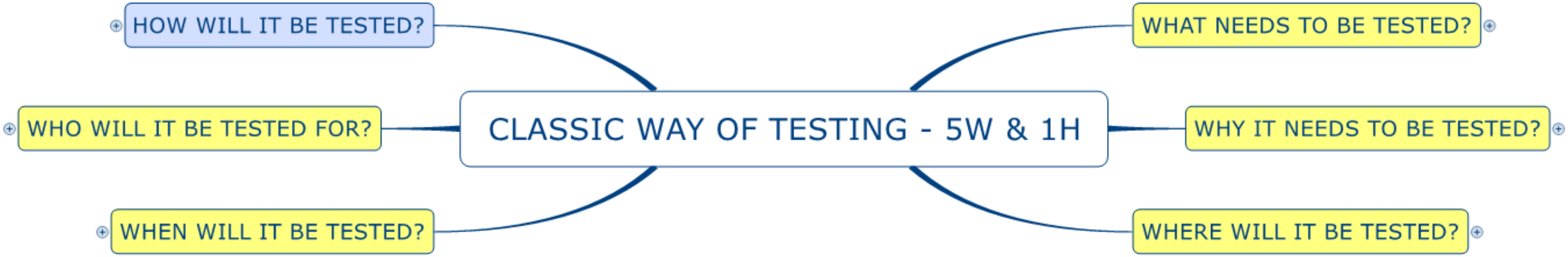
- **Test and re-test are test activities**
  Testing shows system failures.
  Re-testing proves, that the defect has been corrected.

- **Debugging and correcting defects are developer activities**
  Through debugging, developers can reproduce failures, investigate the state of programs and find the corresponding defect in order to correct it.

Traditional Testing in SDLC



Basic Testing Questions



CLASSIC WAY OF TESTING - 5W & 1H

HOW WILL IT BE TESTED?

WHO WILL IT BE TESTED FOR?

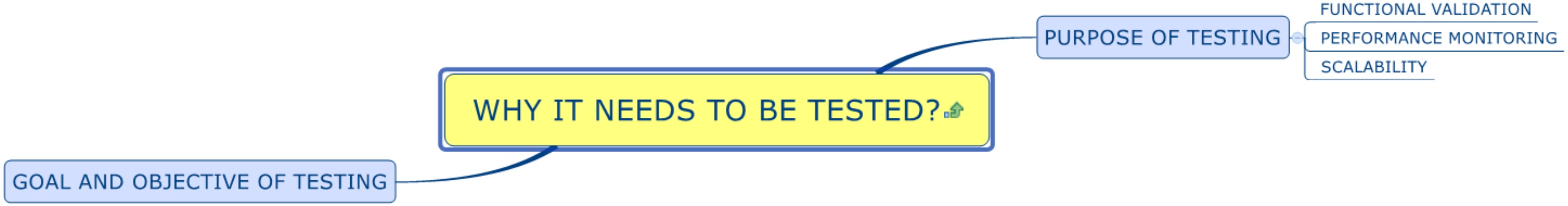
WHEN WILL IT BE TESTED?

WHAT NEEDS TO BE TESTED?

WHY IT NEEDS TO BE TESTED?

WHERE WILL IT BE TESTED?

Why does it need to be tested?



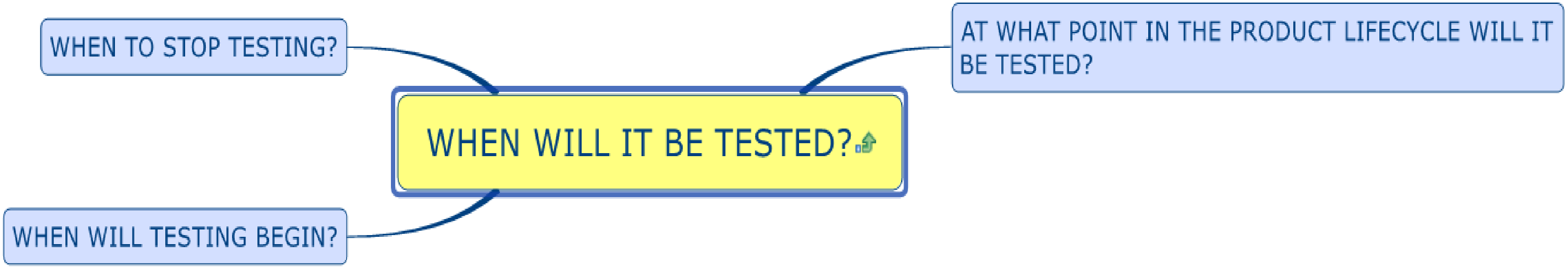
WHY IT NEEDS TO BE TESTED?

PURPOSE OF TESTING

- FUNCTIONAL VALIDATION
- PERFORMANCE MONITORING
- SCALABILITY

GOAL AND OBJECTIVE OF TESTING

When will it be tested?



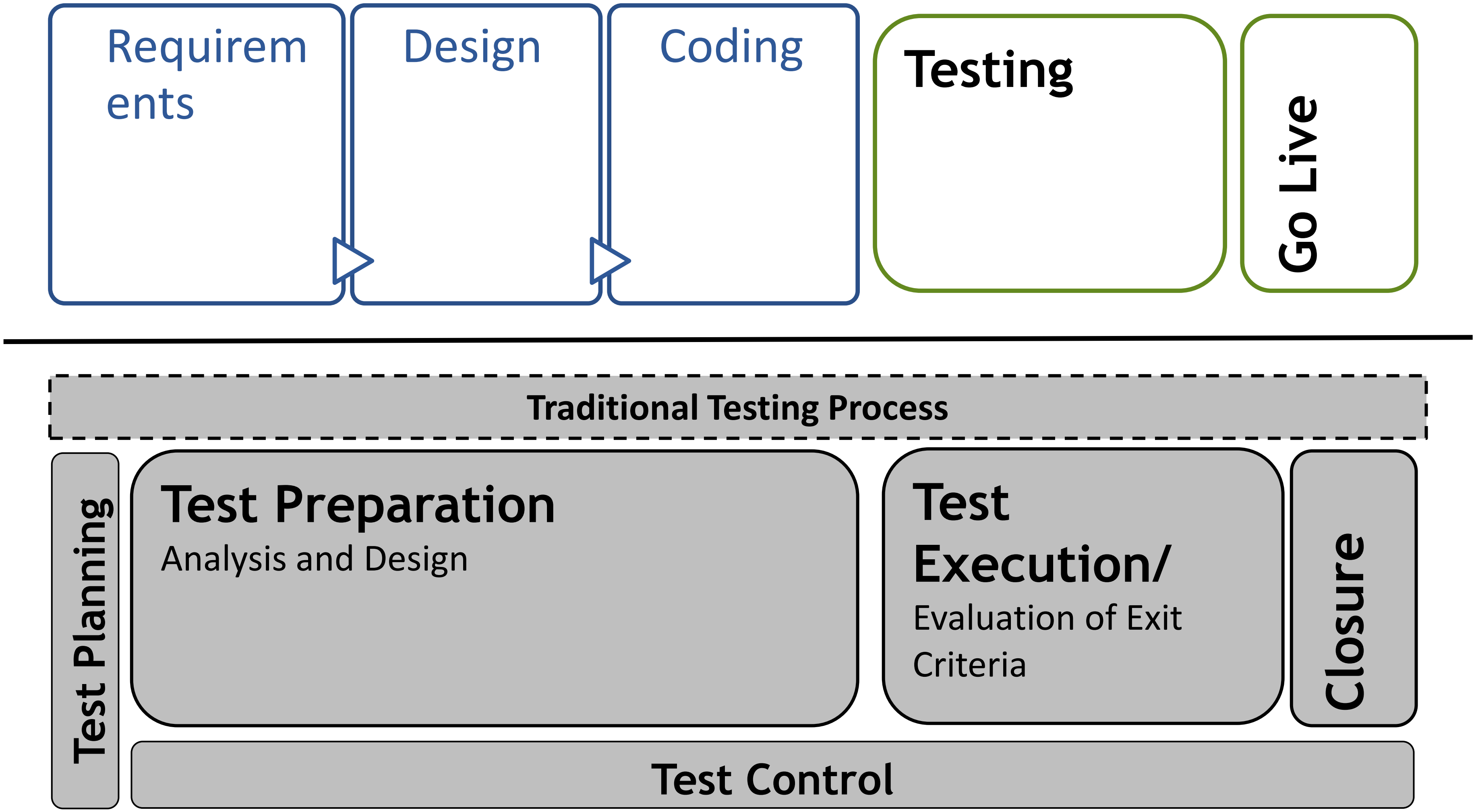
WHEN WILL IT BE TESTED?

WHEN TO STOP TESTING?

WHEN WILL TESTING BEGIN?

AT WHAT POINT IN THE PRODUCT LIFECYCLE WILL IT BE TESTED?

Testing Process



Requirem ents

Design

Coding

Testing

Go Live

Traditional Testing Process

Test Planning

Test Preparation

Analysis and Design

Test Execution/

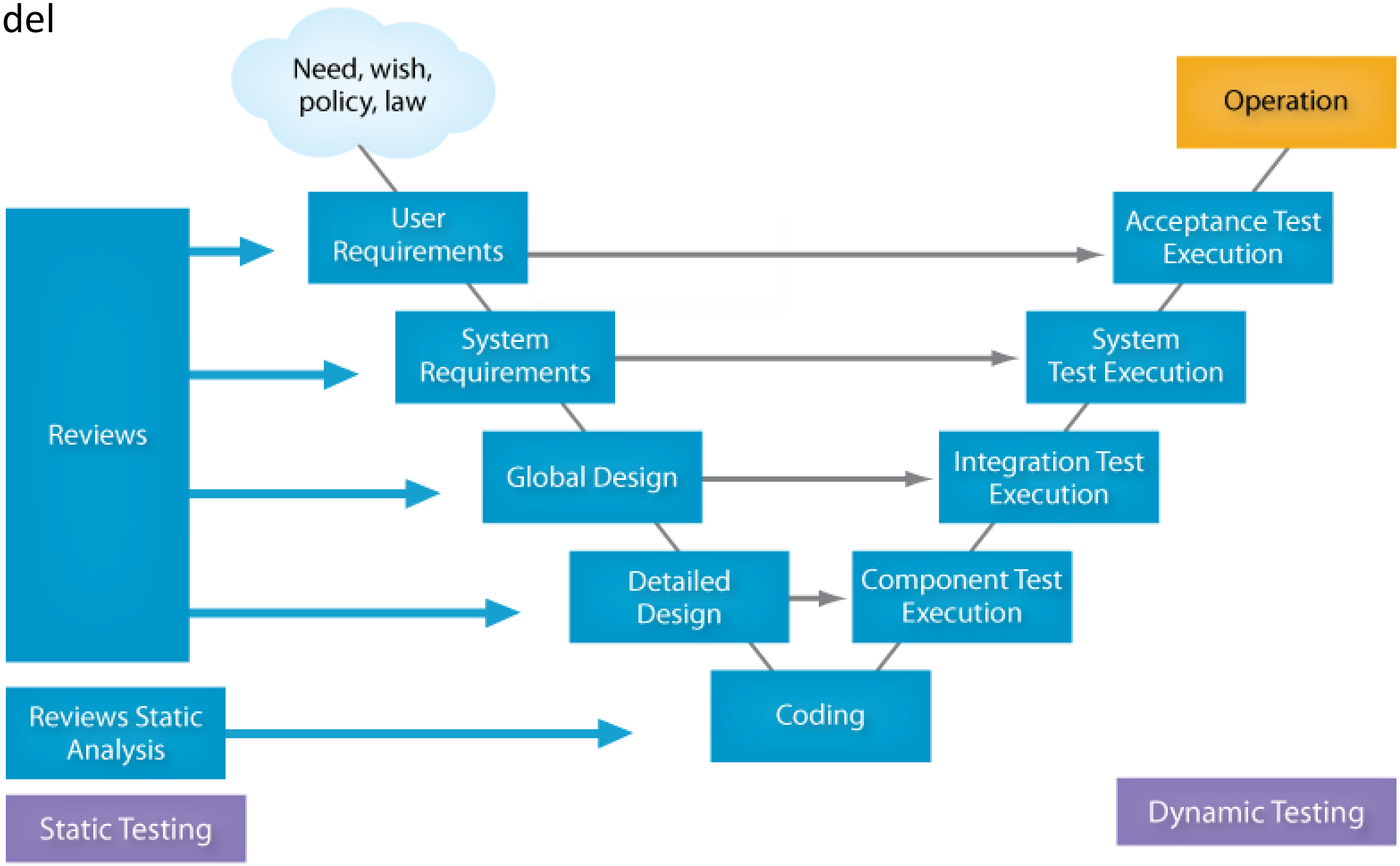
Evaluation of Exit Criteria

Closure

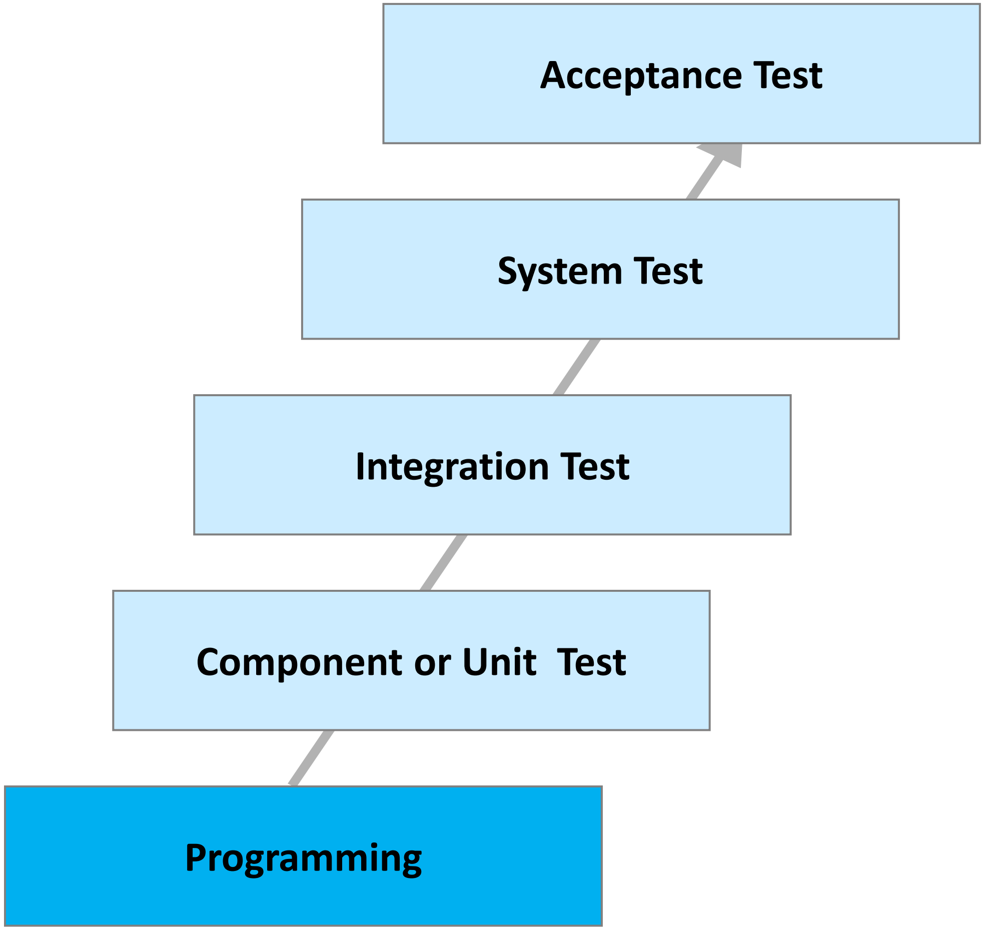
Test Control

V Model

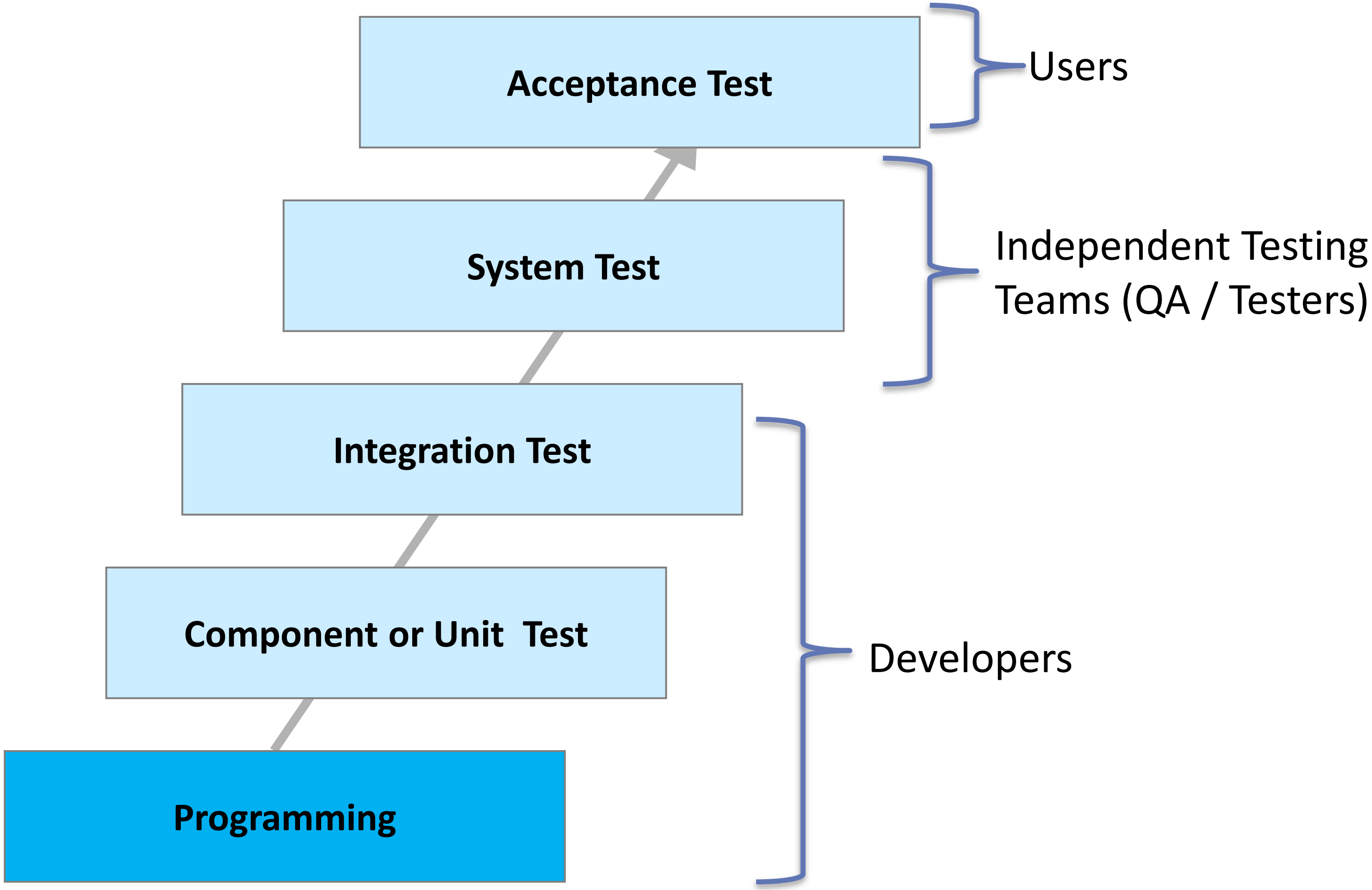
V-Model



Test Levels


Acceptance Test
System Test
Integration Test
Component or Unit Test
Programming

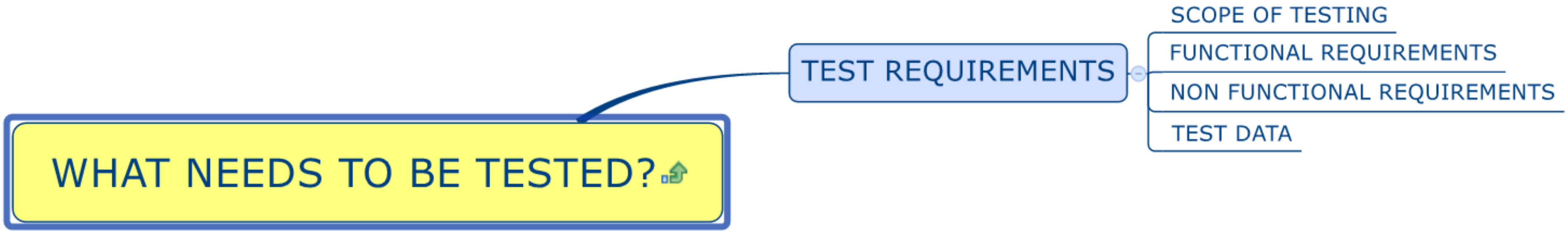
Traditional Testing – Roles



Traditional Test Plan

- Introduce the milestone and schedule tracker

What needs to be tested?



WHAT NEEDS TO BE TESTED?

TEST REQUIREMENTS

SCOPE OF TESTING

FUNCTIONAL REQUIREMENTS

NON FUNCTIONAL REQUIREMENTS

TEST DATA

What is to be tested?

- What is a requirement ?
- What is a test scenario and a test case
- Introduce Test Scenario and Test Case Template with an Exercise

Quality Attributes

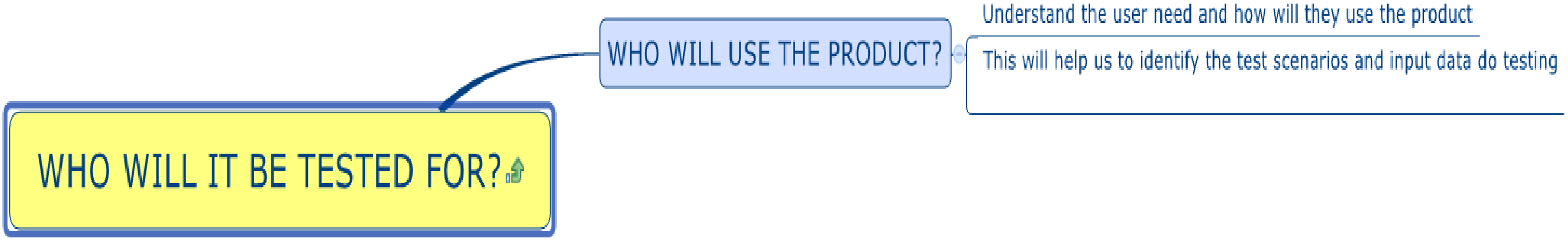
Software quality

- **according to ISO/IEC 9126 software quality consists of:**
  - Functionality — functional Q-attributes
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

  } non-functional Q-attributes

- **Types of Quality Assurance (QA):**
  - **constructive** activities to prevent defects, e.g. through appropriate methods of software engineering
  - **analytical** activities for finding defects, e.g. through testing leading to correcting defects and preventing failures, hence increasing the software quality.

Who will it be tested for?



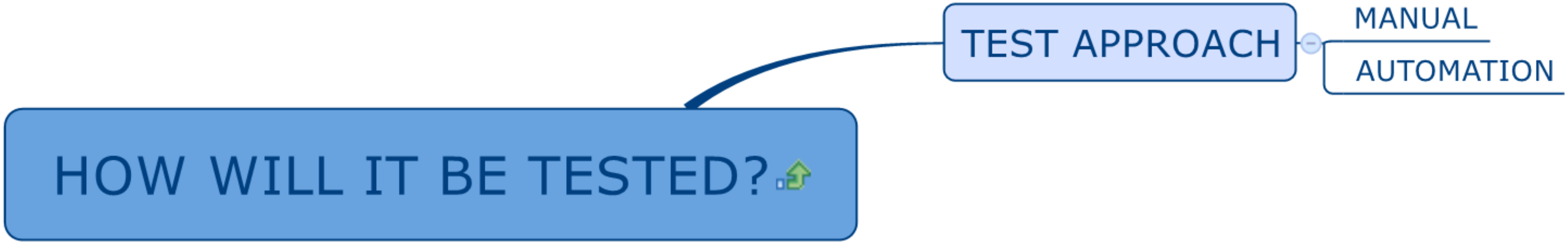
WHO WILL IT BE TESTED FOR?

WHO WILL USE THE PRODUCT?

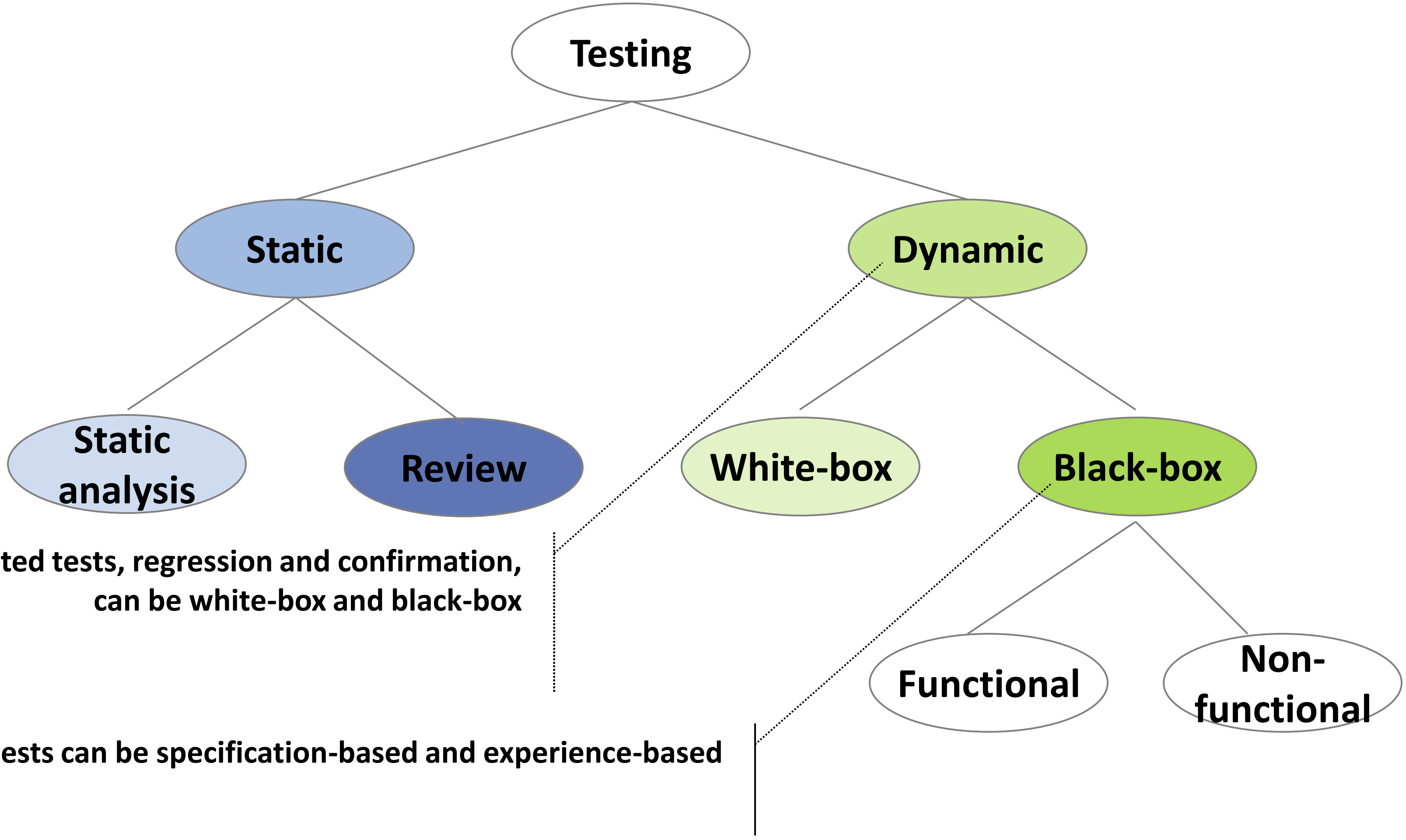
Understand the user need and how will they use the product

This will help us to identify the test scenarios and input data do testing

How will it be tested?



Test Approach



Change-related tests, regression and confirmation, can be white-box and black-box

Black-box tests can be specification-based and experience-based

TEST DESIGN TECHNIQUE

Test Design Techniques - Static

Reviews

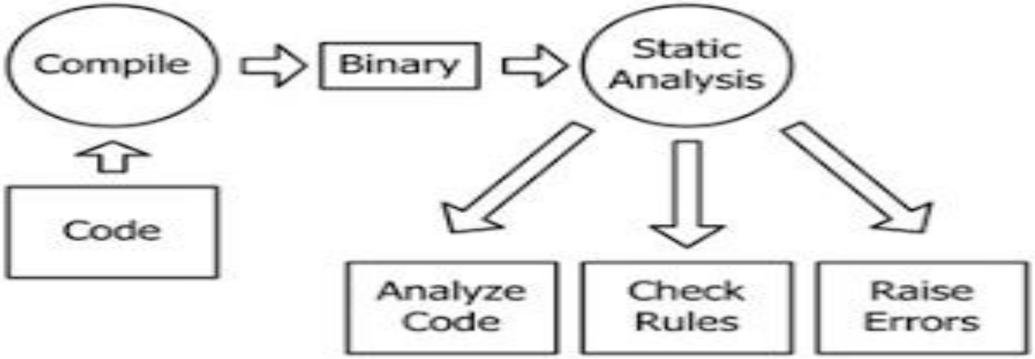
Static Analysis

UNDERSTAND SHIFT LEFT

WHEN TO TEST AND SHIFT LEFT

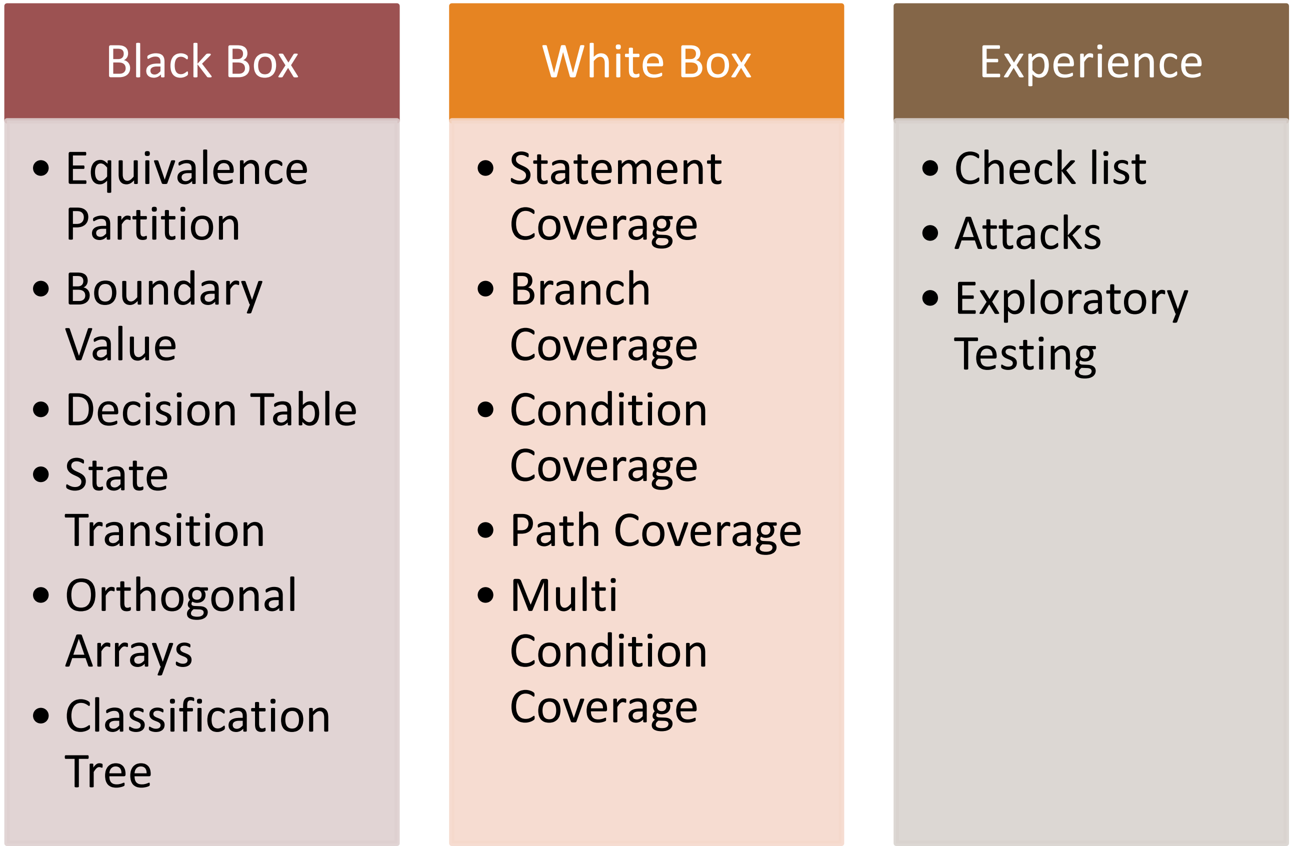
"Shift Left" Testing Strategy

- Static Review
- Static Code Analysis
- Requirements Review



PRACTICAL DYNAMIC TEST DESIGN TECHNIQUE

Test Design Techniques - Dynamic



Black Box

- Equivalence Partition
- Boundary Value
- Decision Table
- State Transition
- Orthogonal Arrays
- Classification Tree

White Box

- Statement Coverage
- Branch Coverage
- Condition Coverage
- Path Coverage
- Multi Condition Coverage

Experience

- Check list
- Attacks
- Exploratory Testing

Test Design Techniques - Dynamic

Black Box

- Equivalence Partition
- Boundary Value
- Decision Table
- State Transition
- Orthogonal Arrays
- Classification Tree

White Box

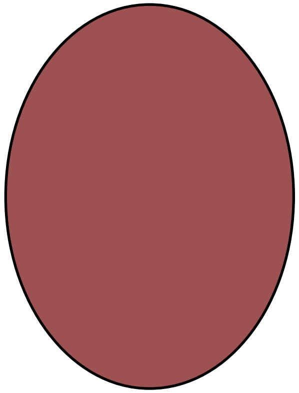
- Statement Coverage
- Branch Coverage
- Condition Coverage
- Path Coverage
- Multi Condition Coverage

Experience

- Check list
- Attacks
- Exploratory Testing

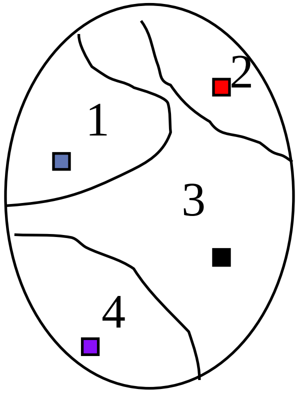
Equivalence Partitioning

- Input domain is usually too large for exhaustive testing.
- Partition input domain into a finite number of sub-domains for the selection of test inputs.
- Each sub-domain is known as an equivalence class and serves as a source of at least one test input.



Input domain

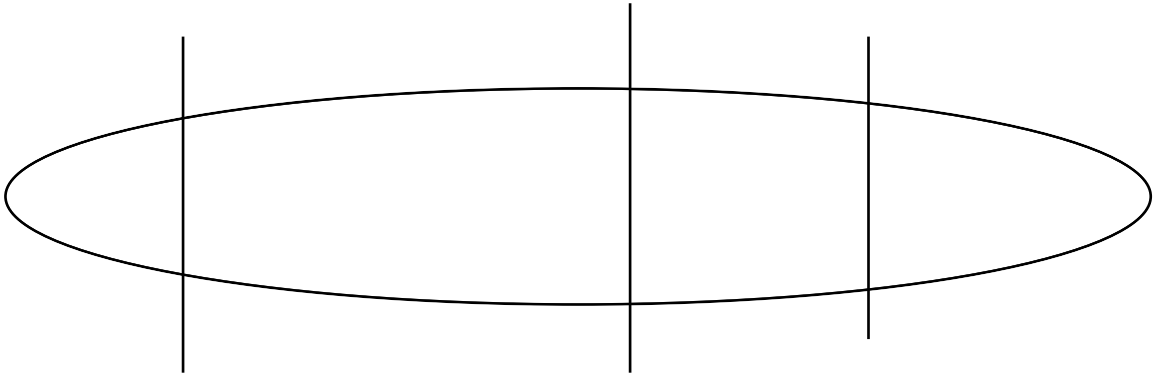
Too many test inputs.



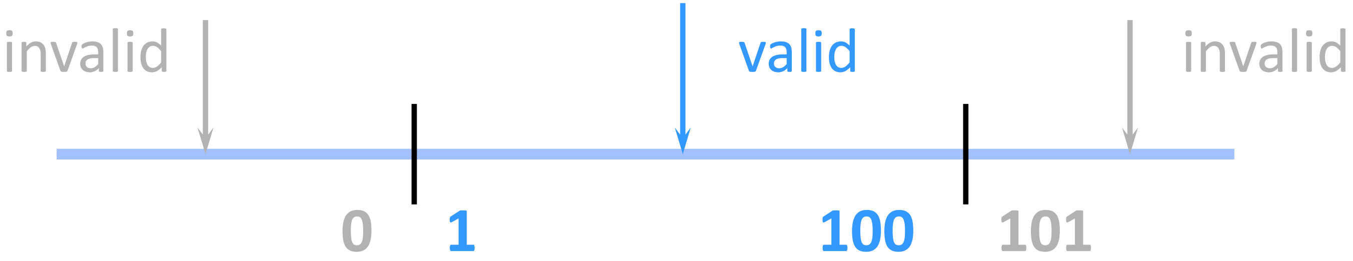
Input domain partitioned into four sub-domains.

Four test inputs, one selected from each sub-domain.

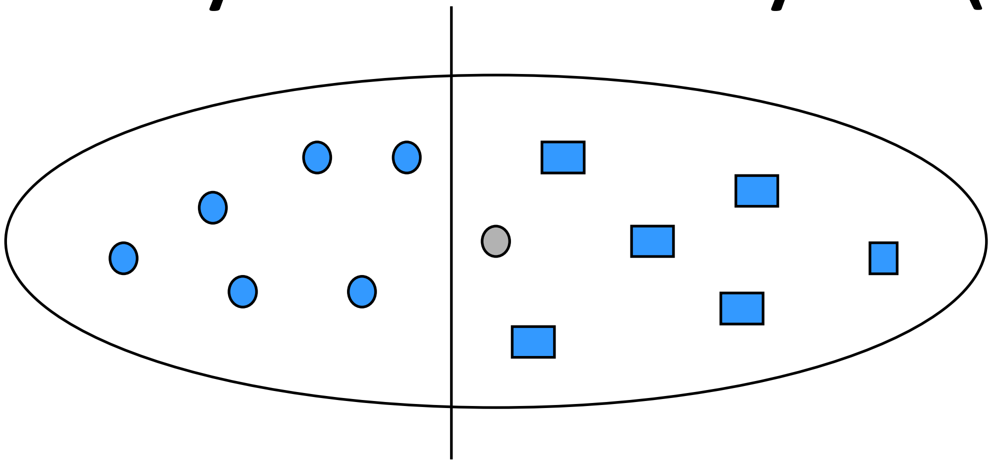
Equivalence partitioning (EP)



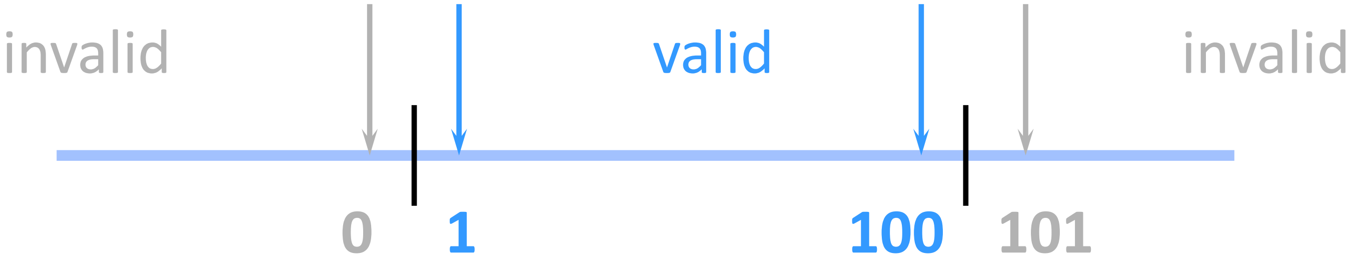
- divide (partition) the inputs, outputs, etc. into areas which are the same (equivalent)
- assumption: if one value works, all will work
- one from each partition better than all from one



Boundary value analysis (BVA)



- faults tend to lurk near boundaries
- good place to look for faults
- test values on both sides of boundaries



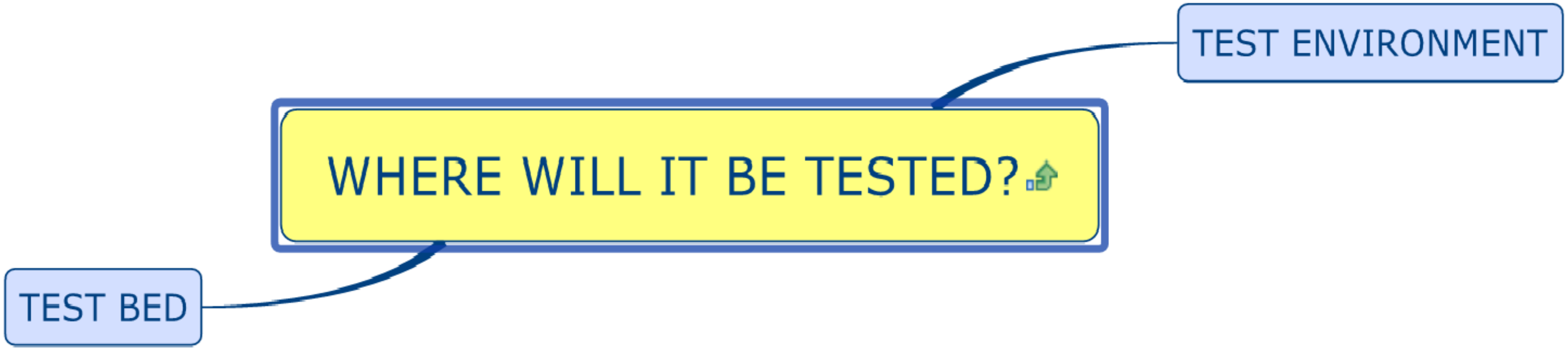
invalid valid invalid

0 1 100 101

Group Activity

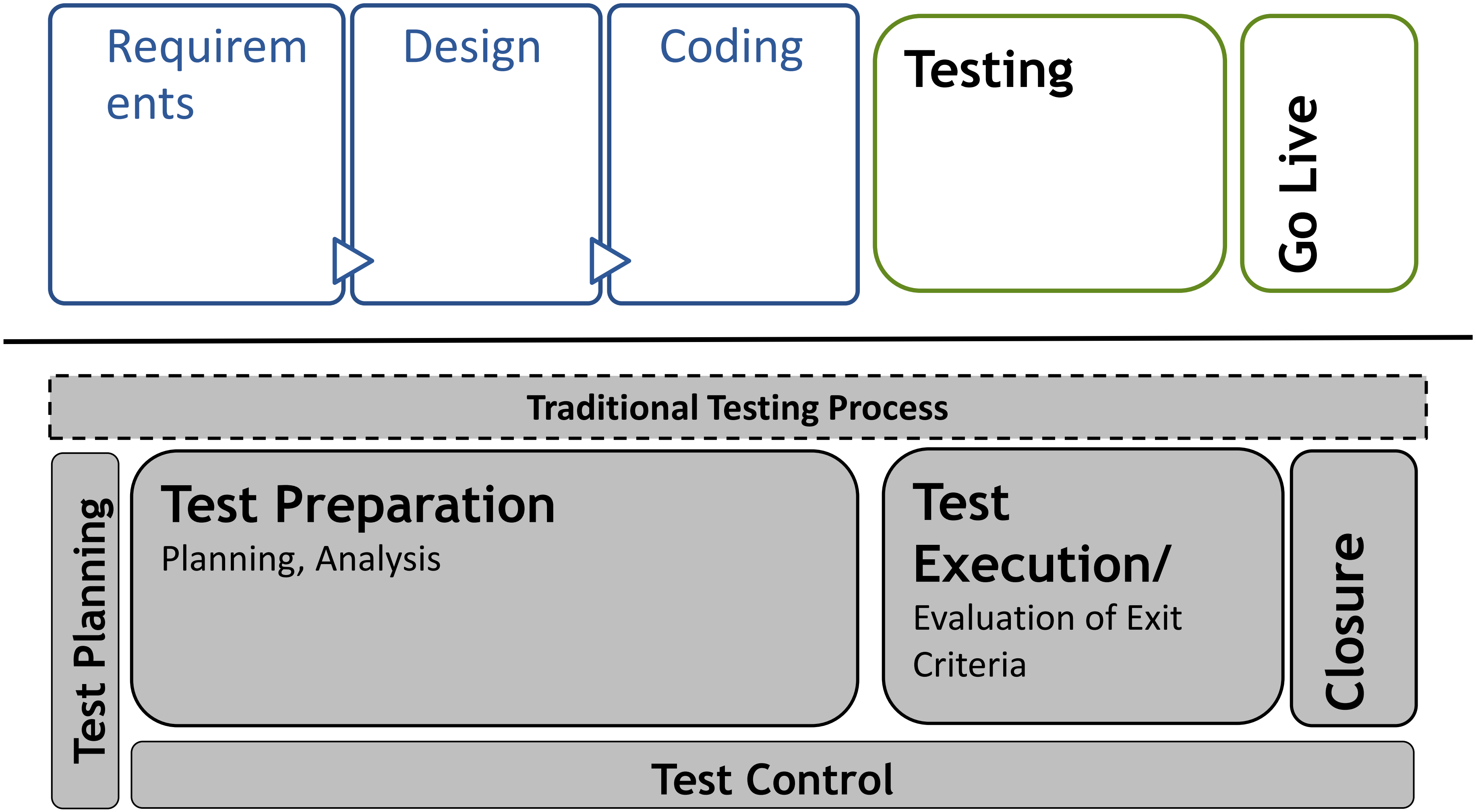
- Case Study – Soft Pac(Review, Create Test Scenarios and Test Cases)
- Requirement document to be shared with participants

Where will it be tested?



TEST EXECUTION AND REGRESSION TESTING

Testing Process



Requirements

Design

Coding

Testing

Go Live

Traditional Testing Process

Test Planning

Test Preparation

Planning, Analysis

Test Execution/

Evaluation of Exit Criteria

Closure

Test Control

Test Execution

- Reporting
- Defects
- Regression Testing

Group Activity

- Case Study – Soft Pac Test Execution

(Defect Reporting)

(Share defect report templates)

SoftPac system for test execution

http://114.79.134.116:3000/SoftPac/login

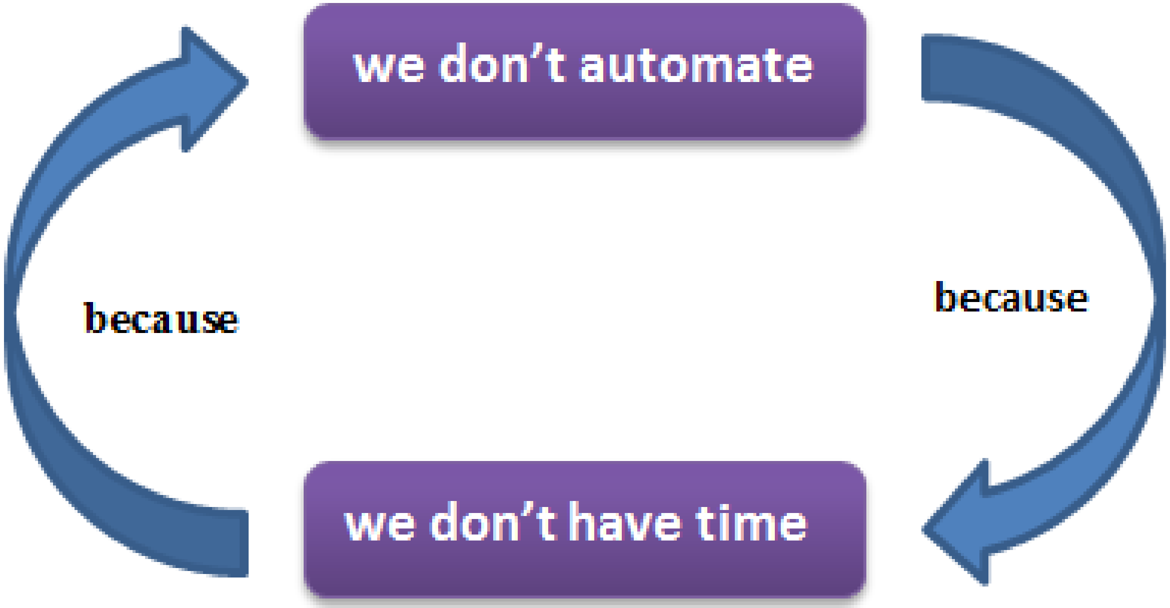
How much Testing is enough?

How much testing is enough?

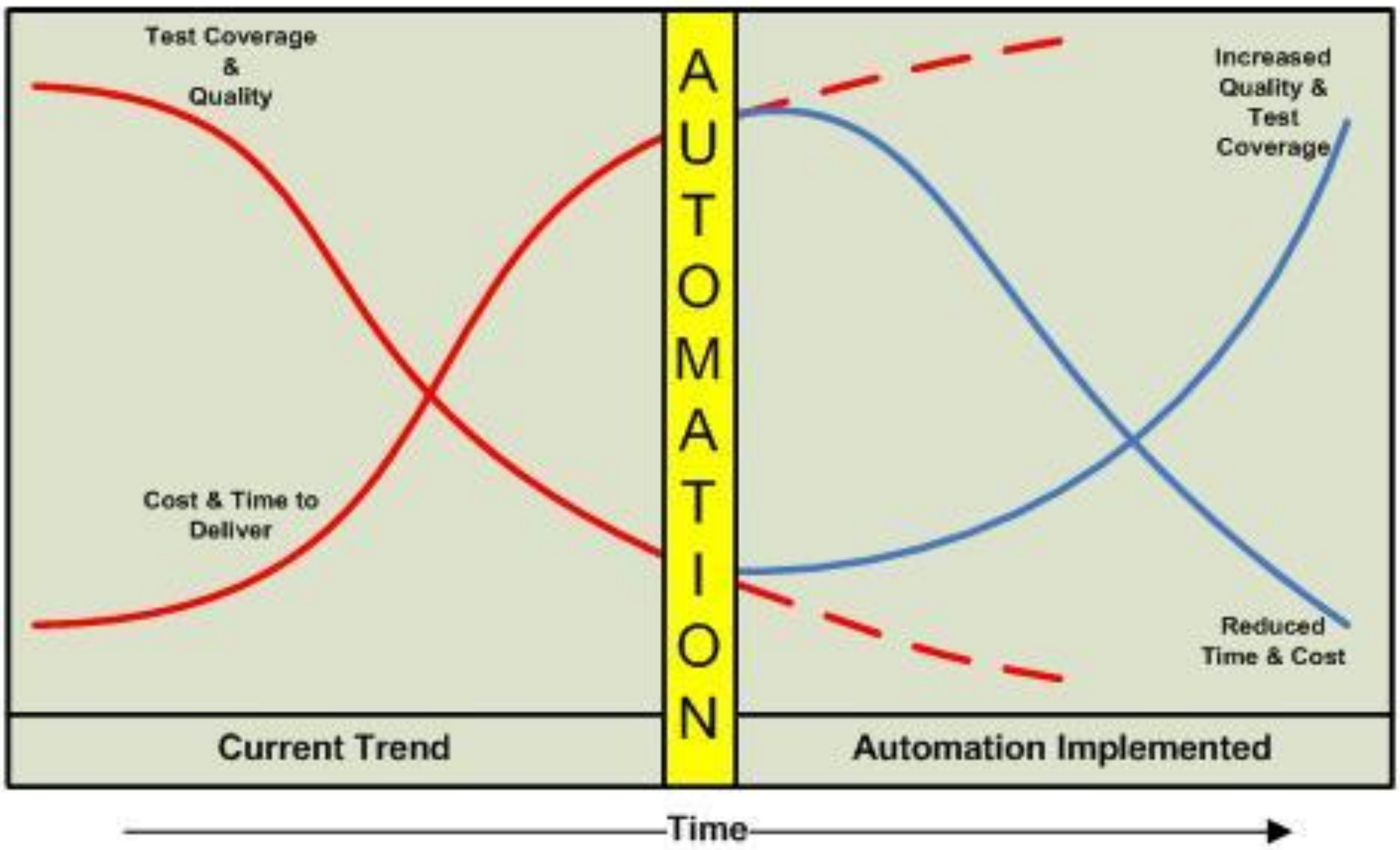
- **Exit criteria**
  not finding (any more) defects is not an appropriate criterion to stop testing activities. Other metrics are needed to adequately reflect the quality level reached.
  - **Risk based testing**
    Levels of risk determine the extent of testing carried out, i.e. liability for damages in case of failure, probability of failures occurring, economic and project related aspects.
  - **Time and budget testing**
    The amount of resources available (personnel, time and budget) might determine the extent of testing efforts.

Test Automation

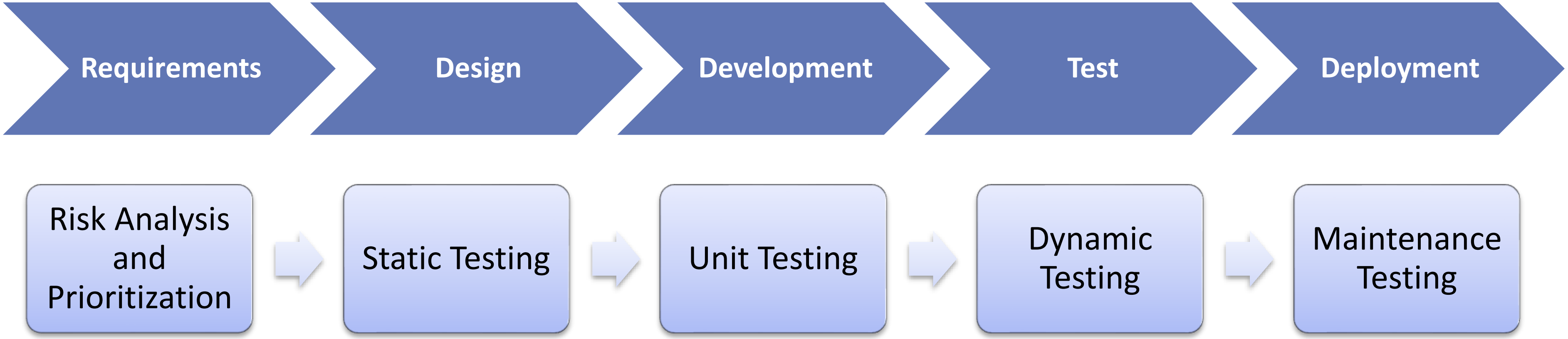
- *T in*
  A
- *F vel*
  te ration
- *A 'e*
  p
  - , , ,



Test Automation



SDLC and Early Testing



Requirements

Design

Development

Test

Deployment

Risk Analysis and Prioritization

Static Testing

Unit Testing

Dynamic Testing

Maintenance Testing

Traditional Test Plan

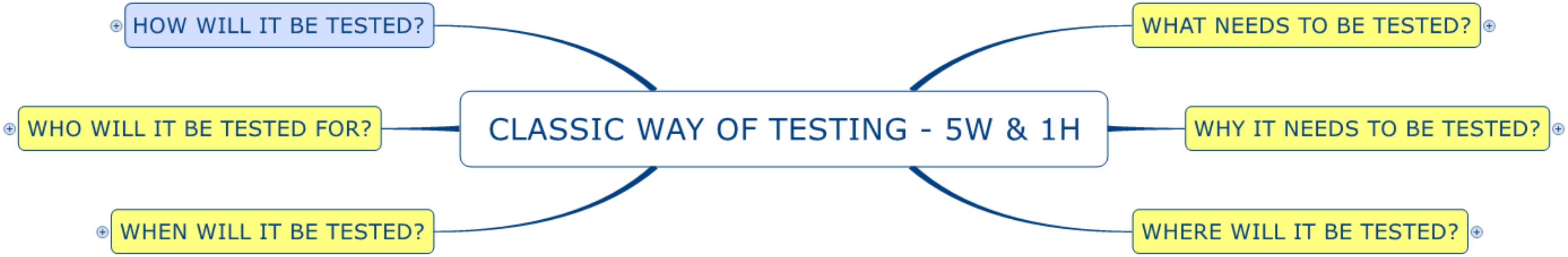
- Introduce the Traditional Test Plan template

Group Activity

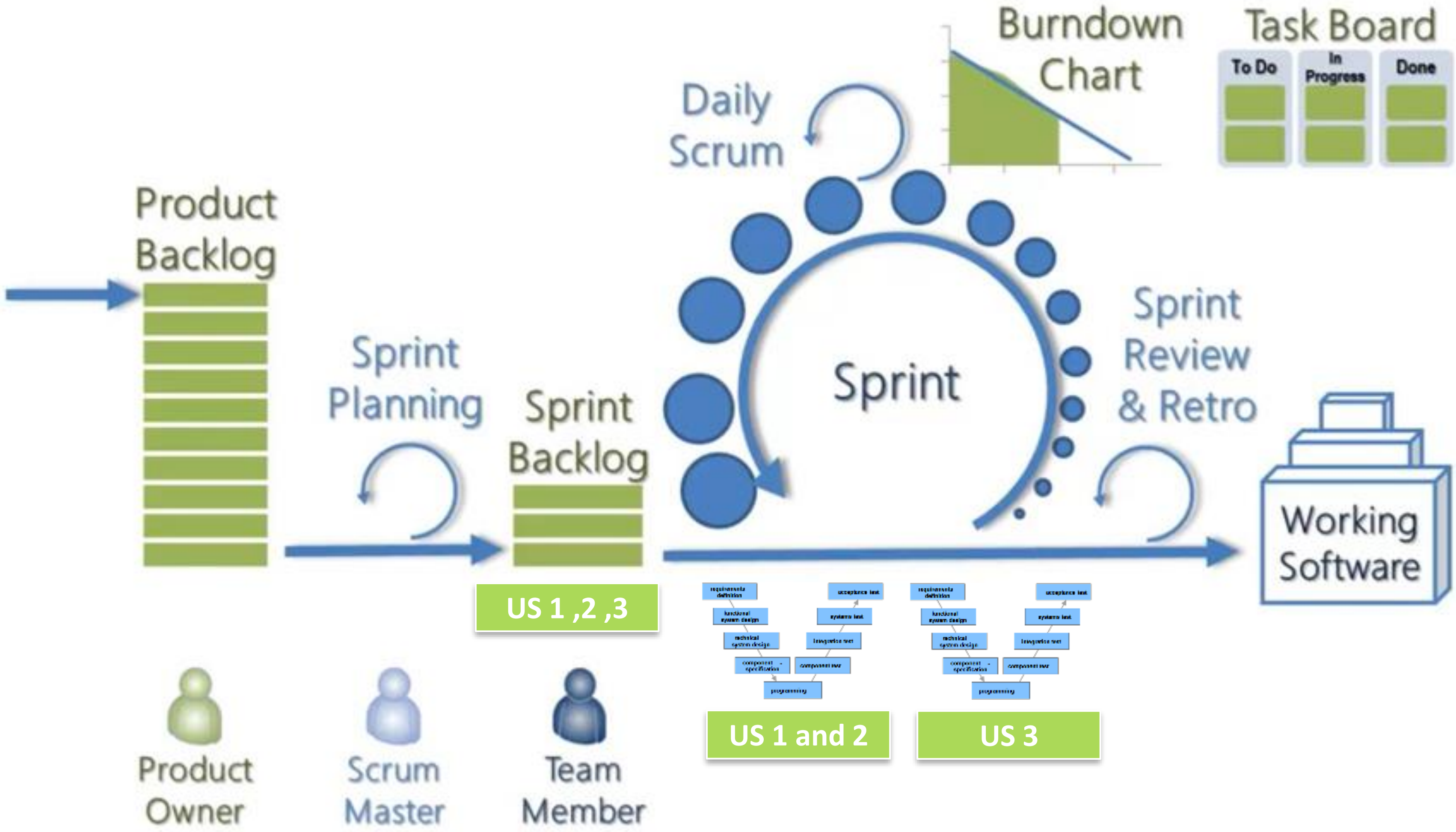
- Case Study – ORHMS Execution

WHAT IS AGILE TESTING

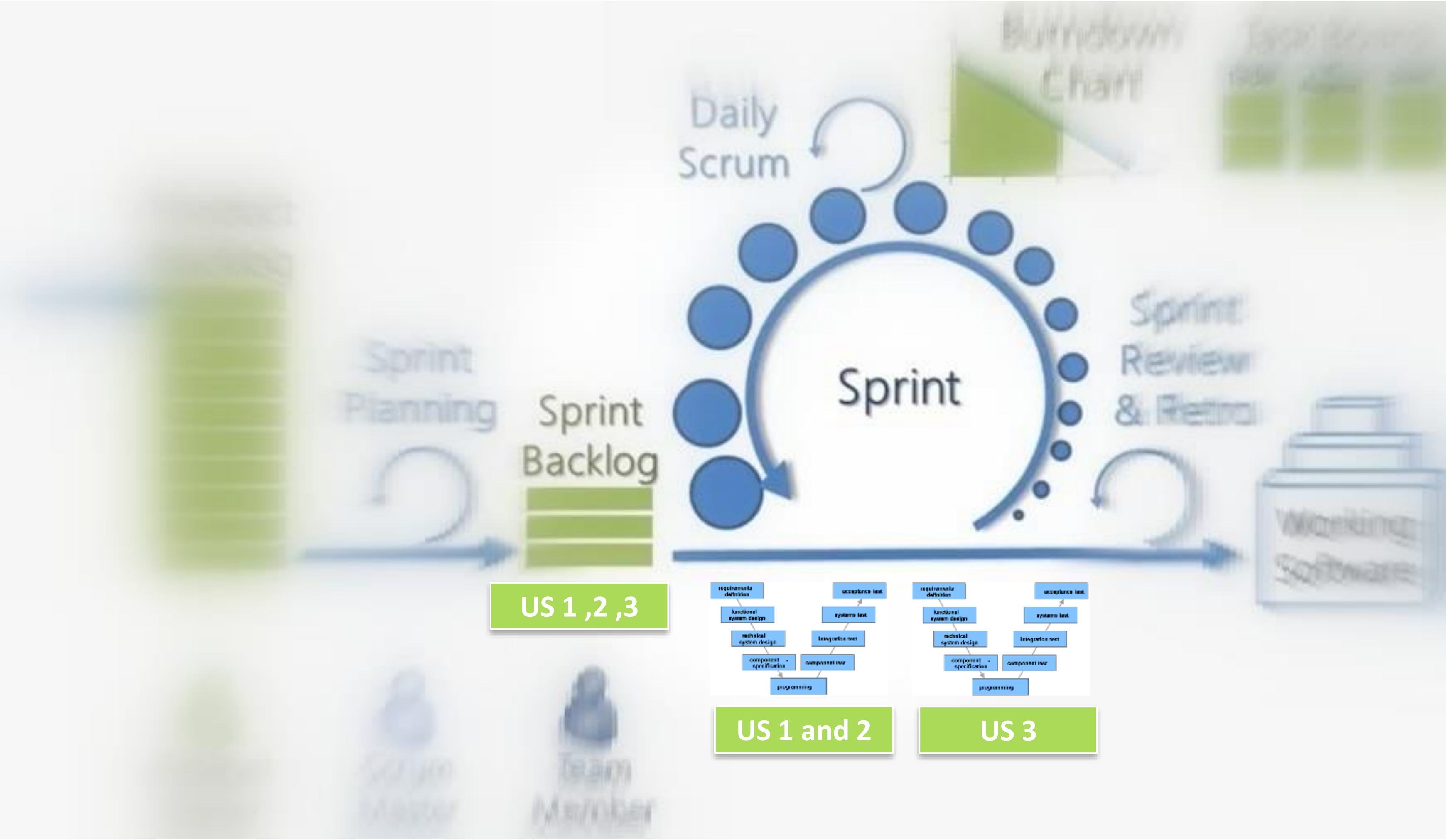
Testing in Agile ?



‘V’ Inside Sprint



‘V’ Inside Sprint



What could be challenges in Agile Testing?

Challenges

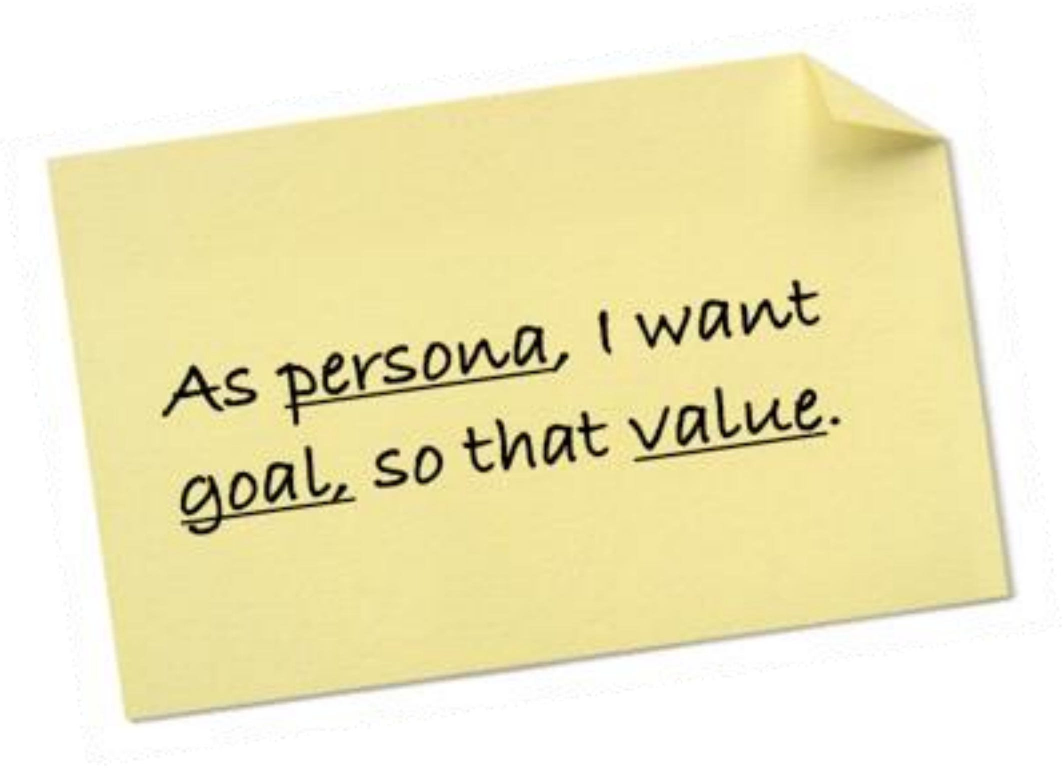
- Time – How do I test in a sprint which is only 2 weeks?
- Timing – When should I start to test?
- Regression – The AUT is changing significantly in every Sprint/Drop, too much regression pack building up.
- Automation – Need for Automation, which tools to use? Will there be RoI?
- Testers vs. Developer Mindset – We are still not a cohesive unit and there is a divide between Testers and Developers.
- Communication – Testers not in synch with project progress
- Documentation – What should I document? Who decides?
- Estimation – How can I estimate and commit for delivery when I have insufficient test oracle and test basis?
- Independence – I feel I will loose the advantage of the Independent Testing while testing as a whole team approach.
- "ility" Testing – When should I do Performance and Usability testing?

USER STORIES (WHAT TO TEST?)

What Is A User Story?

- User stories are a **simple** way of describing and tracking project requirements.
- User stories describe **user observable features** the system needs to provide.
- User stories keep us focused on **delivering value** to end users, rather than focus on internal tasks.
- **Prioritized** by the **customer**

User Story Format



As persona, I want goal, so that value.

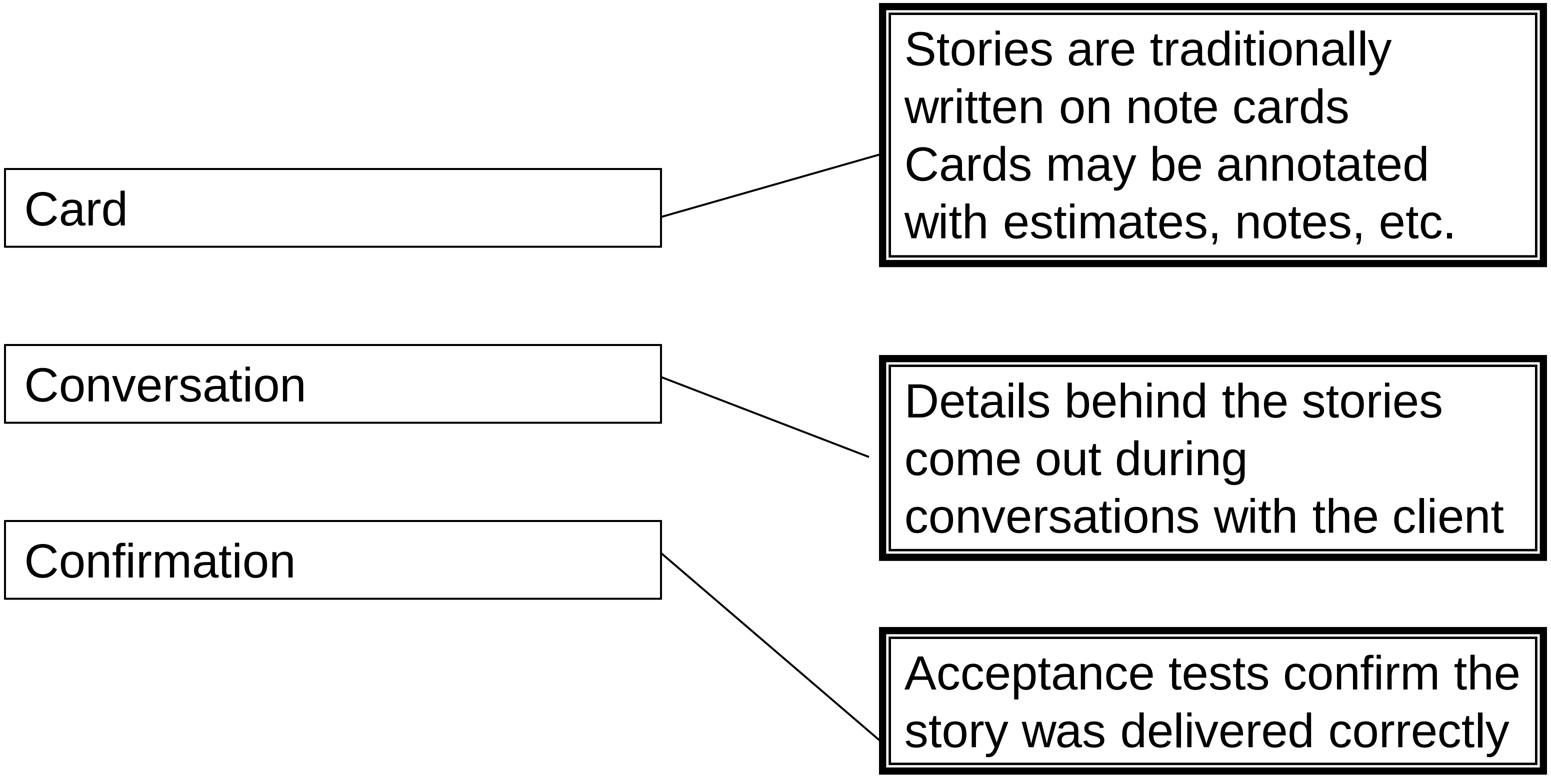
User Story Example

- As a traveler, I want to check in for my upcoming itinerary, so that I save time at the airport.

- As a customer service agent, I want to be able to select seats for an itinerary, so that I can offer my customers their preferred seating.

- As a pricing analyst, I want to generate revenue reports for markets I manage, so I can make good pricing decisions

3 C’s Of User Story



Card

Stories are traditionally written on note cards
Cards may be annotated with estimates, notes, etc.

Conversation

Details behind the stories come out during conversations with the client

Confirmation

Acceptance tests confirm the story was delivered correctly

Good Stories

- Story Card Format
  - As a _____,
  - I want to be able to _____
  - so that _____ .
- Acceptance Criteria
  - I will know this is done when _____ .
- Key Points to Writing Stories
  - Keep stories short & business language focused
  - Seek a level of granularity that can be completed in a few days
  - Keep stories mutually independent
  - Do not include implementation details
  - Do not stop talking –

I – Independent

N – Negotiable

V – Valuable

E – Estimable

S – Small

T – Testable

Acceptance Criteria

- Remove ambiguity from requirements
- Help product owner answer what is needed in order for story / feature to provide value
- Shared understanding of the story / feature
- Know when to stop adding more functionality to a story
- Aid developers and testers to derive tests

Group Activity

User Stories Review using INVEST)

As a user, I want to convert the movie into MPG format so that I can view it on my mobile phone

I want the software to be developed in VB .NET

As an airline user, I want to get the boarding-pass generated so that I can save time

As a developer, I want to know which of my stories have failing test cases so that I can fix the code

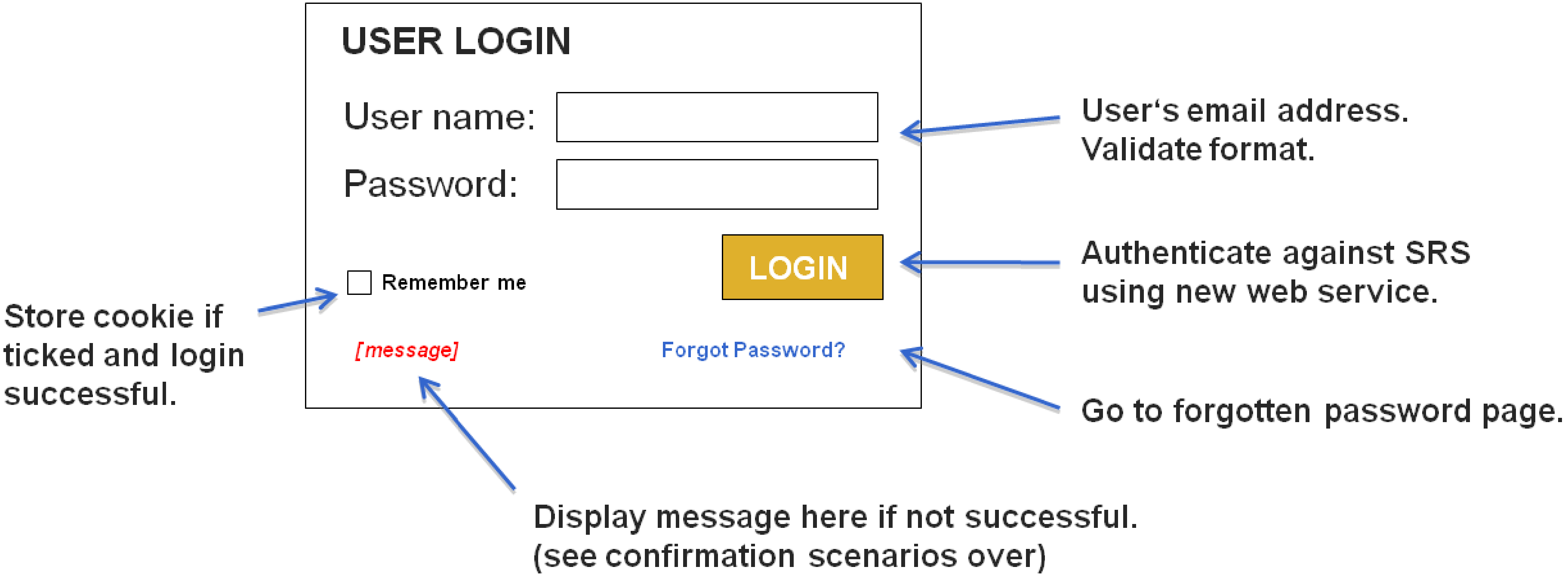
Group Activity

For the user story cards shown in next two slides

1. Review for defects
2. Highlight the importance of SHIFT LEFT

#0001	USER LOGIN	Fibonacci Size #3
As a **[registered user]**, I want to **[log in]**, so I can **[access subscriber content]**.		

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, explain scope/standards.



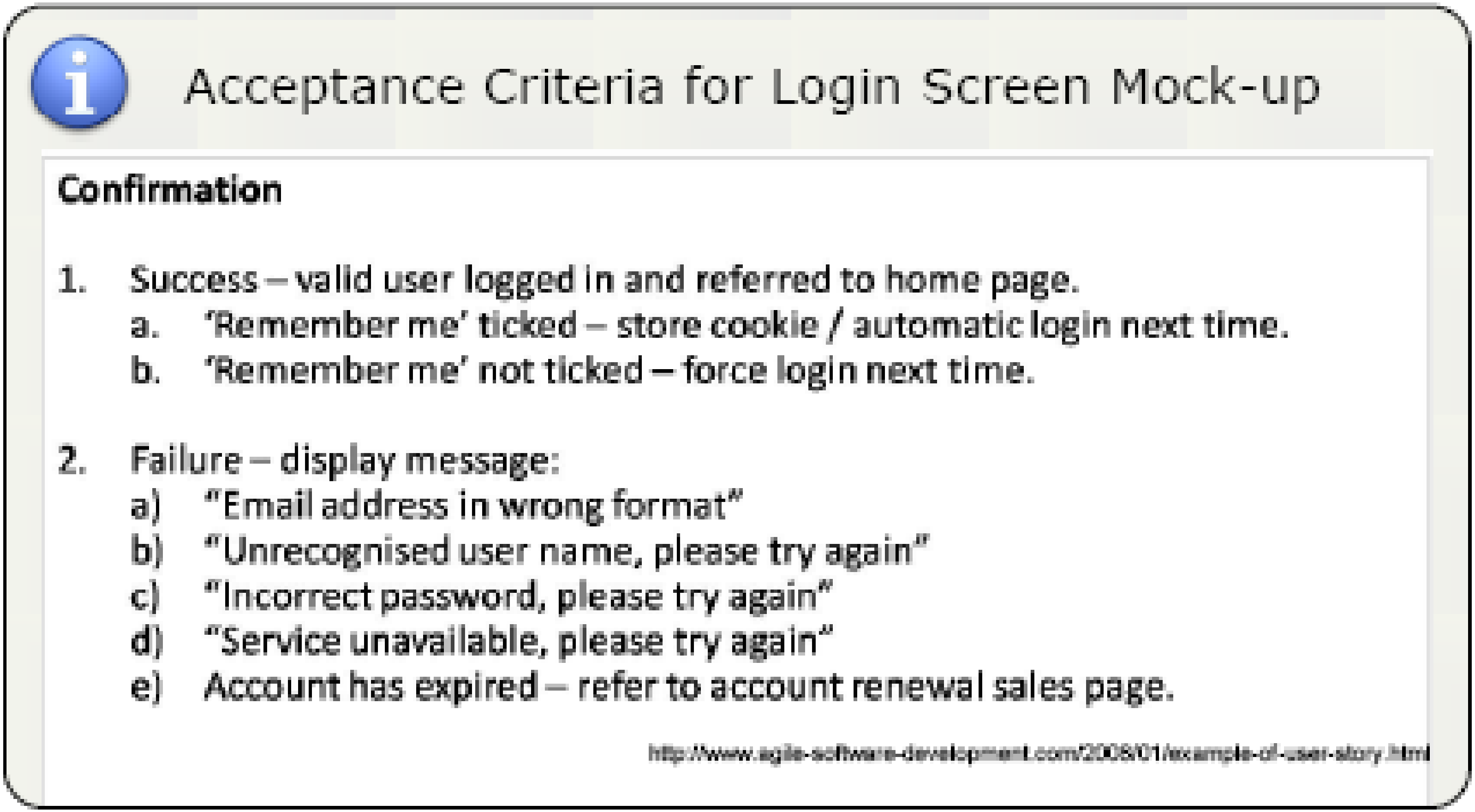
User's email address. Validate format.

Authenticate against SRS using new web service.

Store cookie if ticked and login successful.

Go to forgotten password page.

Display message here if not successful. (see confirmation scenarios over)



Acceptance Criteria for Login Screen Mock-up

Confirmation

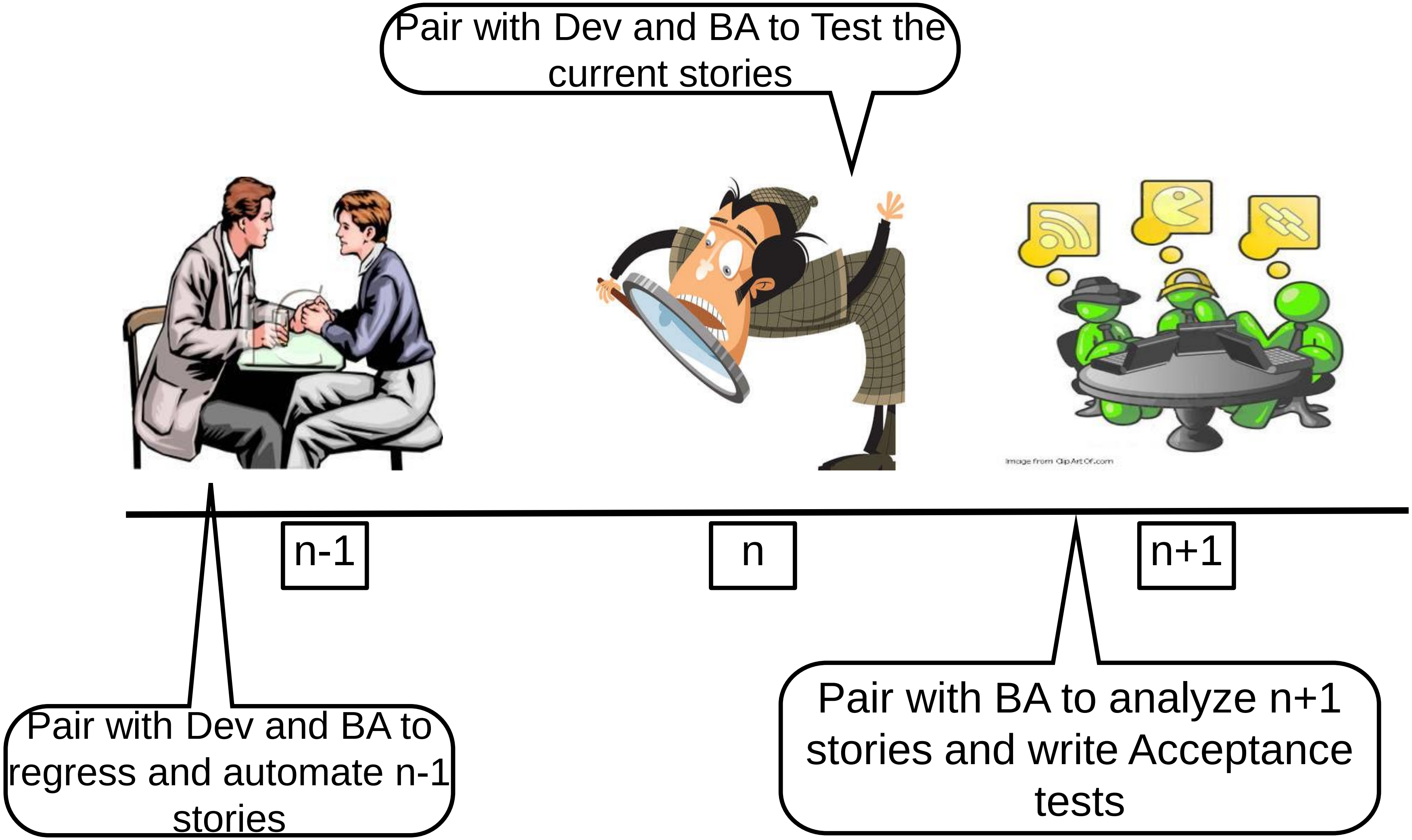
1. Success – valid user logged in and referred to home page.
   a. 'Remember me' ticked – store cookie / automatic login next time.
   b. 'Remember me' not ticked – force login next time.

2. Failure – display message:
   a) "Email address in wrong format"
   b) "Unrecognised user name, please try again"
   c) "Incorrect password, please try again"
   d) "Service unavailable, please try again"
   e) Account has expired – refer to account renewal sales page.

http://www.agile-software-development.com/2008/01/example-of-user-story.html

AGILE PLANNING, TRACKING & MONITORING (WHEN IN THE LIFECYCLE WILL TESTING HAPPEN?)

At any point of time, QA

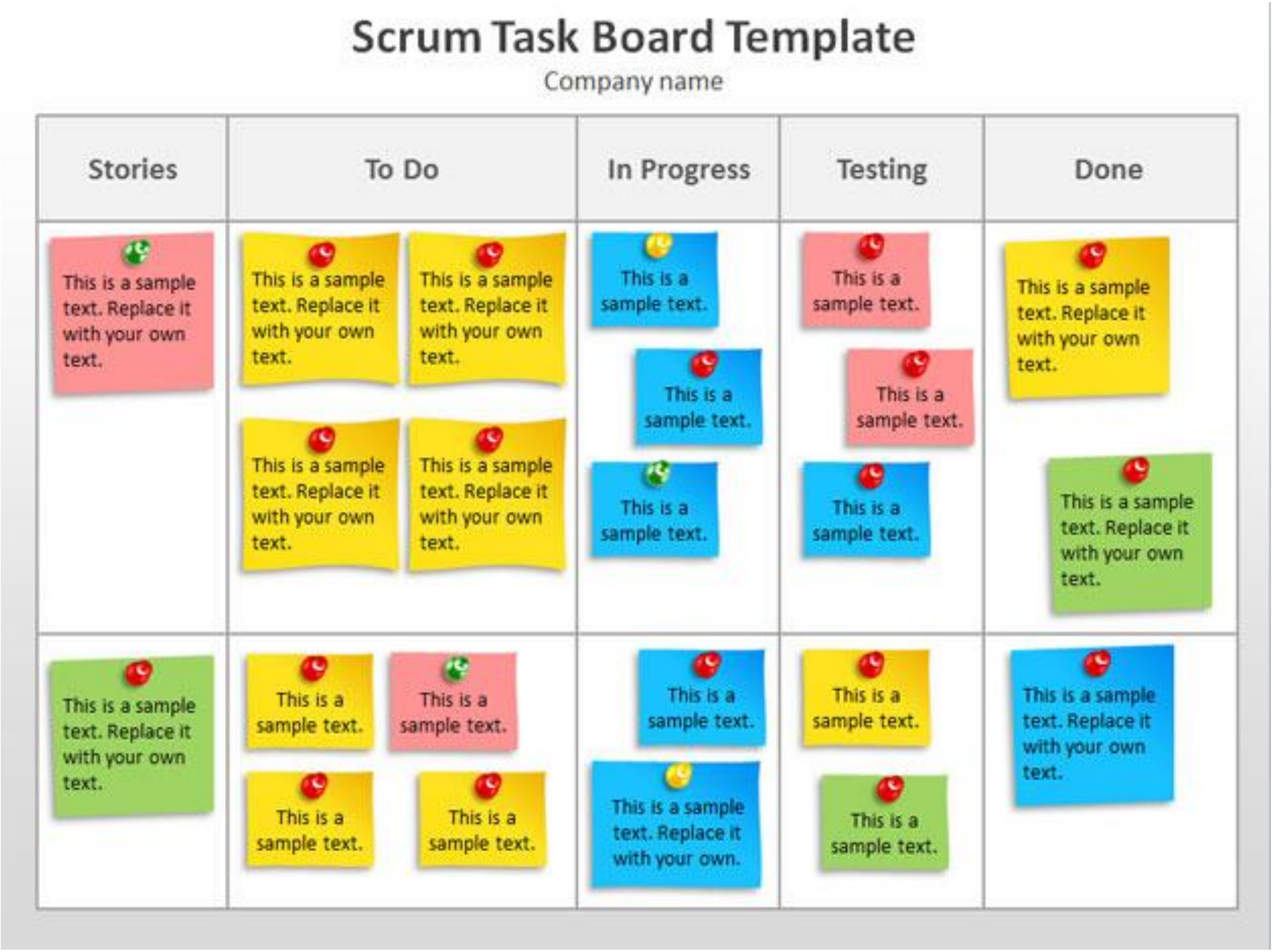


Pair with Dev and BA to Test the current stories

Image from ClipArtOf.com

Pair with Dev and BA to regress and automate n-1 stories

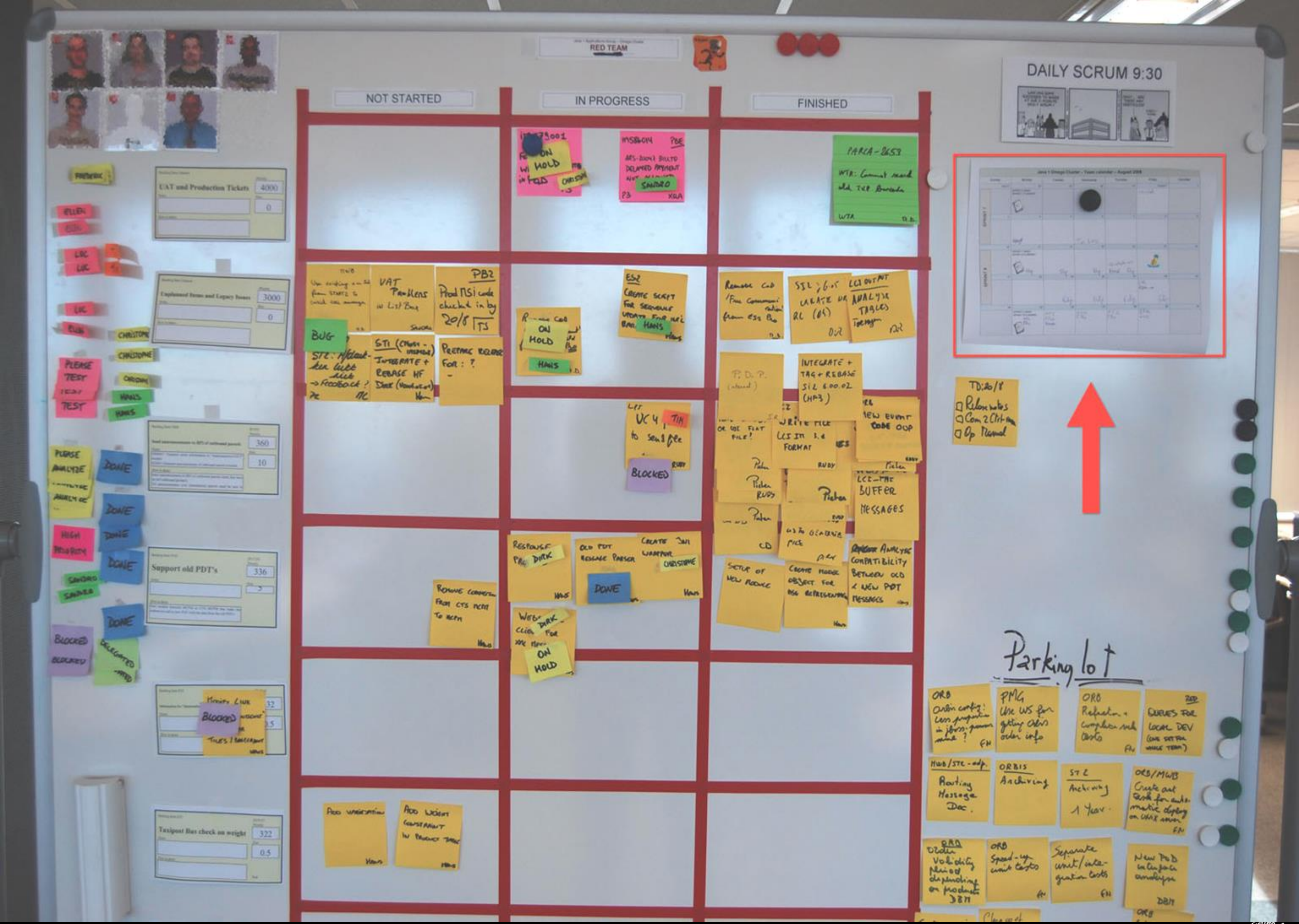
Pair with BA to analyze n+1 stories and write Acceptance tests



Scrum Task Board Template

Company name

Stories	To Do	In Progress	Testing	Done
This is a sample text. Replace it with your own text.	This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text.	This is a sample text. This is a sample text. This is a sample text.	This is a sample text. This is a sample text. This is a sample text.	This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text.
This is a sample text. Replace it with your own text.	This is a sample text. This is a sample text. This is a sample text. This is a sample text.	This is a sample text. This is a sample text. Replace it with your own.	This is a sample text. This is a sample text.	This is a sample text. Replace it with your own text.


RED TEAM
NOT STARTED
IN PROGRESS
FINISHED
DAILY SCRUM 9:30
UAT and Production Tickets
Unplanned Items and Legacy Issues
Support old PDT's
Taxipost Bus check on weight
ON HOLD
BUG
BLOCKED
DONE
PLEASE TEST
PLEASE ANALYZE
HIGH PRIORITY
DELEGATED
Parking lot

	Ready	In Progress	Done
Automation			
Manual Testing			
Development			
Analysis			

	Ready	In Progress	Done
Automation			
Manual Testing			
Development			
Analysis	🟨⬜🟧🟩		

	Ready	In Progress	Done
Automation			
Manual Testing			
Development			
Analysis	⬜ 🟧 🟩	🟨	

	Ready	In Progress	Done
Automation			
Manual Testing			
Development			
Analysis	🟧 🟩	⬜	🟨

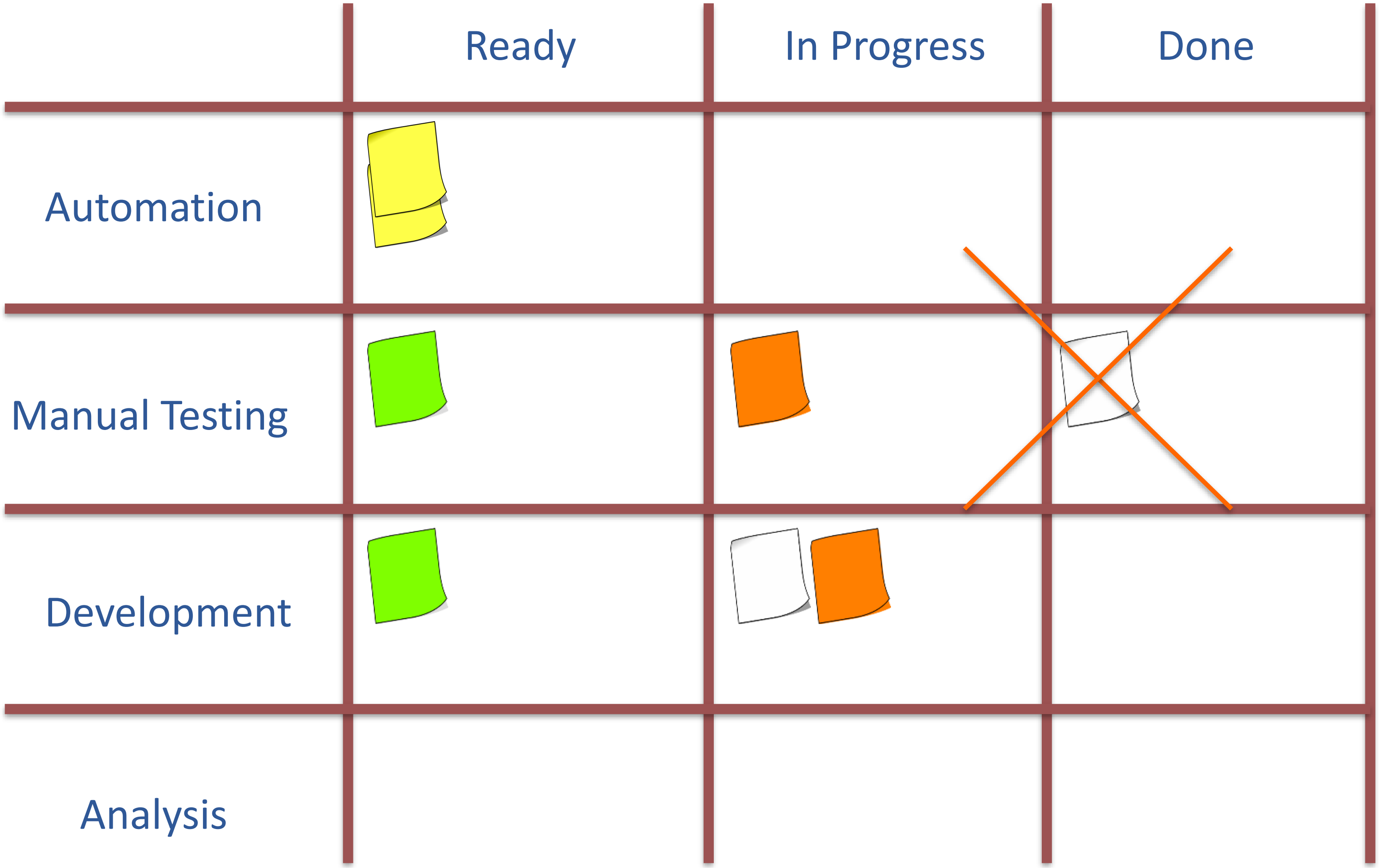
	Ready	In Progress	Done
Automation			
Manual Testing	🟨		
Development	🟨		
Analysis	🟩	🟧	⬜

	Ready	In Progress	Done
Automation			
Manual Testing	⬜	🟨	
Development	🟨		
Analysis	🟩	🟧	

	Ready	In Progress	Done
Automation			
Manual Testing	□	🟨	
Development	□	🟨	
Analysis		🟩	🟧

	Ready	In Progress	Done
Automation			
Manual Testing		⬜	🟨
Development	🟧	⬜	🟨
Analysis			🟩

	Ready	In Progress	Done
Automation			
Manual Testing	🟧	⬜	🟨
Development	🟩	⬜ 🟧	🟨
Analysis			



	Ready	In Progress	Done
Automation	🟨🟨		
Manual Testing	🟩	🟧	~~⬜~~
Development	🟩	⬜ 🟧	
Analysis			

	Ready	In Progress	Done
Automation	🟨🟨		
Manual Testing	🟩	🟧	⬜
Development	🟩	🟧	⬜
Analysis			

	Ready	In Progress	Done
Automation	□	🟨	
Manual Testing		🟩	🟧
Development		🟩	🟧
Analysis			

	Ready	In Progress	Done
Automation	🟧🟧	⬜⬜	🟨🟨
Manual Testing			🟩
Development			🟩
Analysis			

Group Activity

- Create Task Board for tracking progress of your project

WHAT TO TEST – IDENTIFY SCENARIOS

- Using Acceptance Criteria to identify scenarios
- User workflows for identifying scenarios
- Integration points for scenarios

Exercise : Use Post IT notes for creating scenarios

PRACTICAL TEST DESIGN TECHNIQUE

UNDERSTAND SHIFT LEFT IN AGILE

Test Design Techniques - Static

Reviews

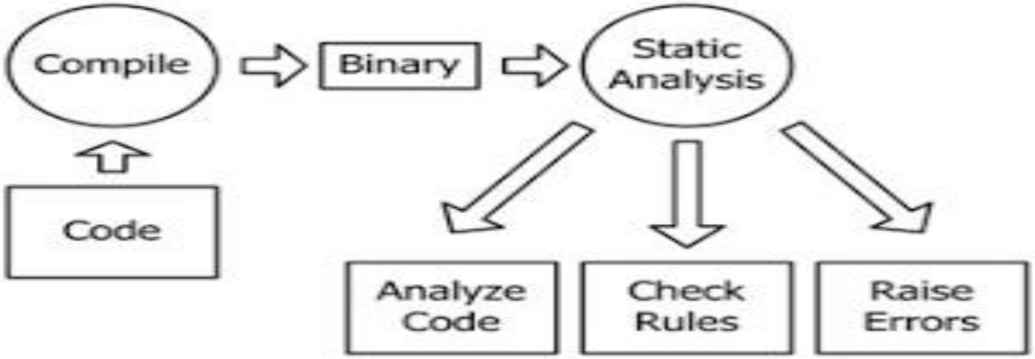
- Technical Review
- Walk Through
- Inspection
- Managerial Review
- Audit
- Peer Review
- Pair Programming
- Pair Testing

Static Analysis

- Programming Rules and Standard
- Control Flow
- Data Flow
- Complexity Analysis

“Shift Left” Testing Strategy

- Static Review
- Static Code Analysis
- Early Test Design using RBT
- Requirements Review, User Stories grooming



PRACTICAL DYNAMIC TEST DESIGN TECHNIQUE – ADVANCED EP

Test Design Techniques - Dynamic

Black Box

- Equivalence Partition
- Boundary Value
- Decision Table
- State Transition
- Orthogonal Arrays
- Classification Tree

White Box

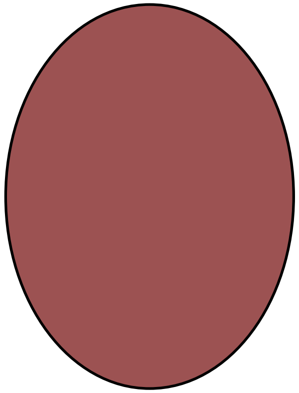
- Statement Coverage
- Branch Coverage
- Condition Coverage
- Path Coverage
- Multi Condition Coverage

Experience

- Check list
- Attacks
- Exploratory Testing

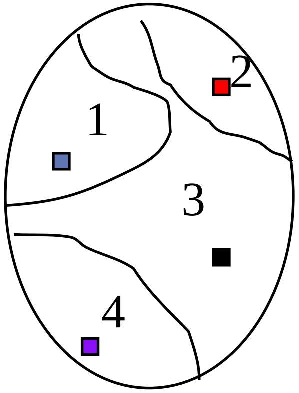
Equivalence Partitioning

- Input domain is usually too large for exhaustive testing.
- Partition input domain into a finite number of sub-domains for the selection of test inputs.
- Each sub-domain is known as an equivalence class and serves as a source of at least one test input.



Input domain

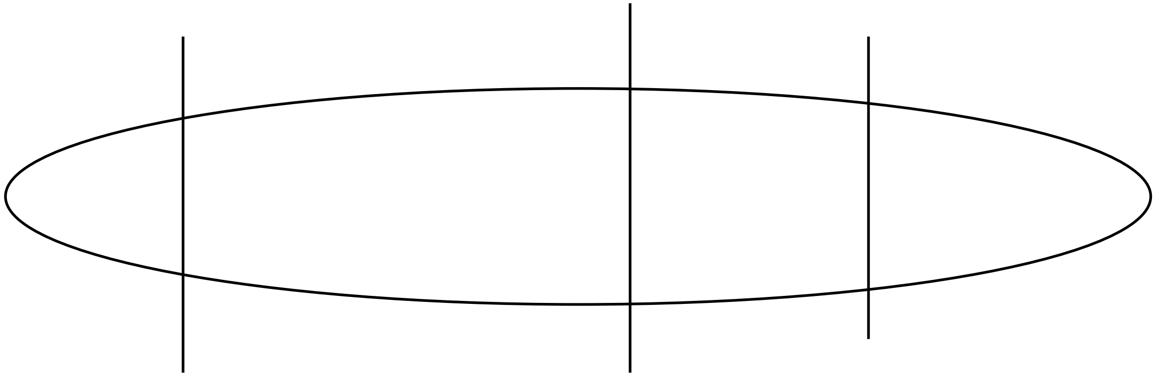
Too many test inputs.



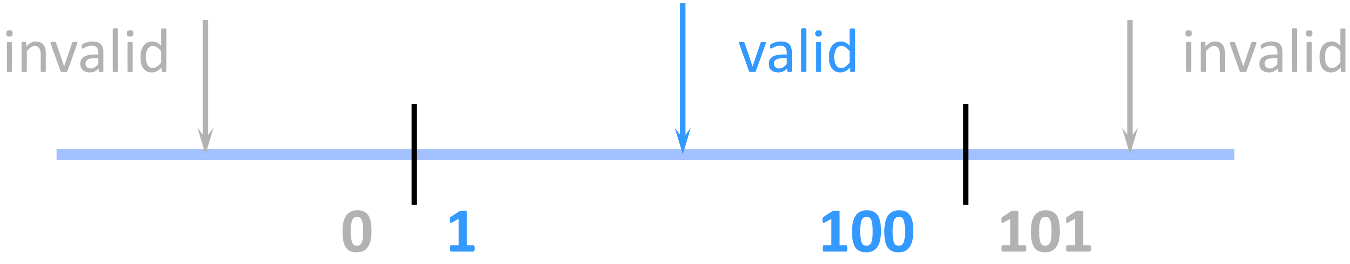
Input domain partitioned into four sub-domains.

Four test inputs, one selected from each sub-domain.

Equivalence partitioning (EP)



- divide (partition) the inputs, outputs, etc. into areas which are the same (equivalent)
- assumption: if one value works, all will work
- one from each partition better than all from one



Equivalence class – Rules in General

- Choosing representatives
  - **any value** within the EC can be a **representative**. Optimal are:
    - typical values (used often)
    - problem values (suspected failures)
    - boundary values (on the edge of the EC)
  - Representatives of **valid EC** may be **combined**
  - Representatives of **invalid EC** may **not be combined**
  - Representatives of **invalid EC** may only be **combined with valid** representatives of **other EC**
  - For test cases, representatives of invalid EC should be combined with always the same values of other valid EC (standard combinations)
  - Choosing representatives implies that the function within the program uses compare operations

Equivalence class partitioning – coverage

- Equivalence class coverage can be used as exit criteria to end testing activities

$$\text{EC - Coverage} = \frac{\text{Number of EC tested}}{\text{Number of EC defined}} * 100\%$$

EP – Exercise

- **Analyzing the specification**
  - A loan application software forwards the application to the relevant Approver, once the applicant enters the loan requested, income entered and duration for the loan chosen.

Assumptions:

- **Loan Amount requested is positive number**
- **Income Entered is a positive number**
- **Duration of the loan can be 1,3,5 and 7 years only**

Find out the Equivalence classes and minimum number of test cases required

EP – Exercise

Blank Page for solution working

EP – Solution-1

Variable	Equivalence Class	Status	Representative
Loan Amount	LA-1: x > 0	valid	100000.00
	LA-2: x <= 0	invalid	-100000.00
	LA-3: x is a special Character	invalid	*
	LA-4 x is non-numerical value	invalid	ATA
Income	In-1: x > 0	valid	200000.00
	In-2: x <= 0	invalid	-200000.00
	In-3: x is a special Character	invalid	$
	In-4 x is non-numerical value	invalid	AGILE
Duration	Du - 1: x = 1	valid	1
	Du - 2: x = 3	valid	3
	Du - 2: x = 5	valid	4
	Du - 2: x = 7	valid	7
	Du: x^{1} {1,3,5,7}	invalid	4
	Du: x non-numerical value	invalid	Testing

EP

Valid Test Cases = 4

Variable	Equivalence Class	Status	Representative	TC1	TC2	TC3	TC4
Loan Amount	LA-1: x > 0	valid	100000.00	√	√	√	√
	LA-2: x <= 0	invalid	-100000.00				
	LA-3: x is a special Character	invalid	*				
	LA-4 x is non-numerical value	invalid	ATA				
Income	In-1: x > 0	valid	200000.00	√	√	√	√
	In-2: x <= 0	invalid	-200000.00				
	In-3: x is a special Character	invalid	$				
	In-4 x is non-numerical value	invalid	AGILE				
Duration	Du - 1: x = 1	valid	1	√			
	Du - 2: x = 3	valid	3		√		
	Du - 2: x = 5	valid	4			√	
	Du - 2: x = 7	valid	7				√
	Du: x ^{1} {1,3,5,7}	invalid	4				
	Du: x non-numerical value	invalid	Testing				

EP

Invalid Test Cases = 8

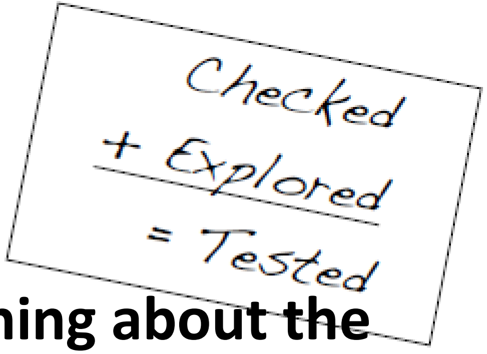
Variable	Equivalence	Status	Representative	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
Loan Amount	LA-1: x >	valid	100000.00				√	√	√	√	√
	LA-2: x <= 0	invalid	-100000.00	√							
	LA-3: x is a special Character	invalid	*		√						
	LA-4 x is non-numerical value	invalid	ATA			√					
Income	In-1: x > 0	valid	200000.00	√	√	√				√	√
	In-2: x <= 0	invalid	-200000.00				√				
	In-3: x is a special Character	invalid	$					√			
	In-4 x is non-numerical value	invalid	AGILE						√		
Duration	Du - 1: x = 1	valid	1	√	√	√	√	√	√		
	Du - 2: x = 3	valid	3								
	Du - 2: x = 5	valid	4								
	Du - 2: x = 7	valid	7								
	Du: x ^{1} {1,3,5,7}	invalid	4							√	
	Du: x non-numerical value	invalid	Testing								√

Total Test Cases 12

Variable	Equivalence Class	Status	Representative	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
Loan Amount	LA-1: x > 0	valid	100000.00	√	√	√	√				√	√	√	√	√
	LA-2: x <= 0	invalid	-100000.00					√							
	LA-3: x is a special Character	invalid	*						√						
	LA-4 x is non-numerical value	invalid	ATA							√					
Income	In-1: x > 0	valid	200000.00	√	√	√	√	√	√	√				√	√
	In-2: x <= 0	invalid	-200000.00								√				
	In-3: x is a special Character	invalid	$									√			
	In-4 x is non-numerical value	invalid	AGILE										√		
Duration	Du - 1: x = 1	valid	1	√				√	√	√	√	√	√		
	Du - 2: x = 3	valid	3		√										
	Du - 2: x = 5	valid	4			√									
	Du - 2: x = 7	valid	7				√								
	Du: x ^{1} {1,3,5,7}	invalid	4											√	
	Du: x non-numerical value	invalid	Testing												√

TEST EXECUTION AND TEST REPORTING

Exploratory Testing



Exploratory Testing involves simultaneously learning about the software under test while designing and executing tests, using feedback from the last test to inform the next.

- Emphasize on the personal freedom and responsibility of the individual tester
- Focus on continually optimize the value of the work by treating
  - test-related learning,
  - test design,
  - test execution, and
  - test result interpretation
- as mutually supportive activities that run in parallel throughout the project.

Exploratory Testing

In contrast with traditional testing, we:

- Design the test as needed
- Execute the test at time of design or reuse it later
- Vary the test as appropriate, whenever appropriate.

Not scripting doesn’t mean not preparing:

- We often design support materials in advance and use them many times throughout testing, such as
  - data sets
  - failure mode lists
  - testing charters

Exploratory Testing - Charters

Traditional testing is documentation-centric with written Test Plans, Test Strategies, Test Cases, and Test Procedures.

Exploratory Testing involves far less documentation. We don't document each and every test case. Instead, we write charters: simple statements of the information that we hope to discover through exploration.

One way of expressing charters is with the simple template:

Explore *area*

With *resources, constraints, tools, etc.*

To discover *information*

Some charters are broad: "Explore the system with typical usage scenarios to discover how it works."

The charters should NOT be too General or too Specific

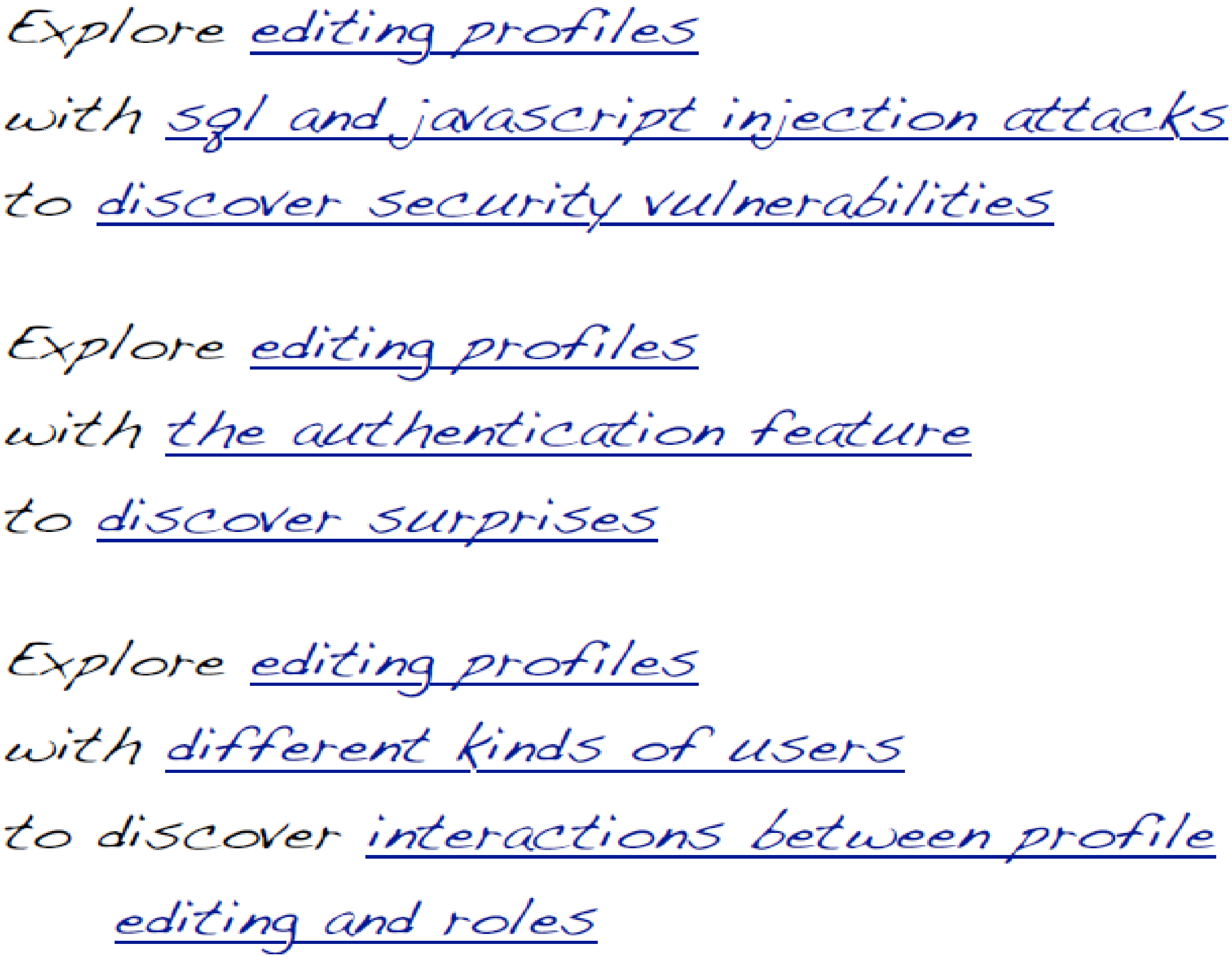
Exploratory Testing - Charters

For example, imagine we have a story like:

As a user, I want to update my personal information on my profile so that my public profile stays up to date.

And we might have charters like:

Exploratory Testing - Charters



Explore editing profiles
with sql and javascript injection attacks
to discover security vulnerabilities

Explore editing profiles
with the authentication feature
to discover surprises

Explore editing profiles
with different kinds of users
to discover interactions between profile
editing and roles

Practical Case Study

- Prepare the plan and charters for Agile case study
- Sprint 1

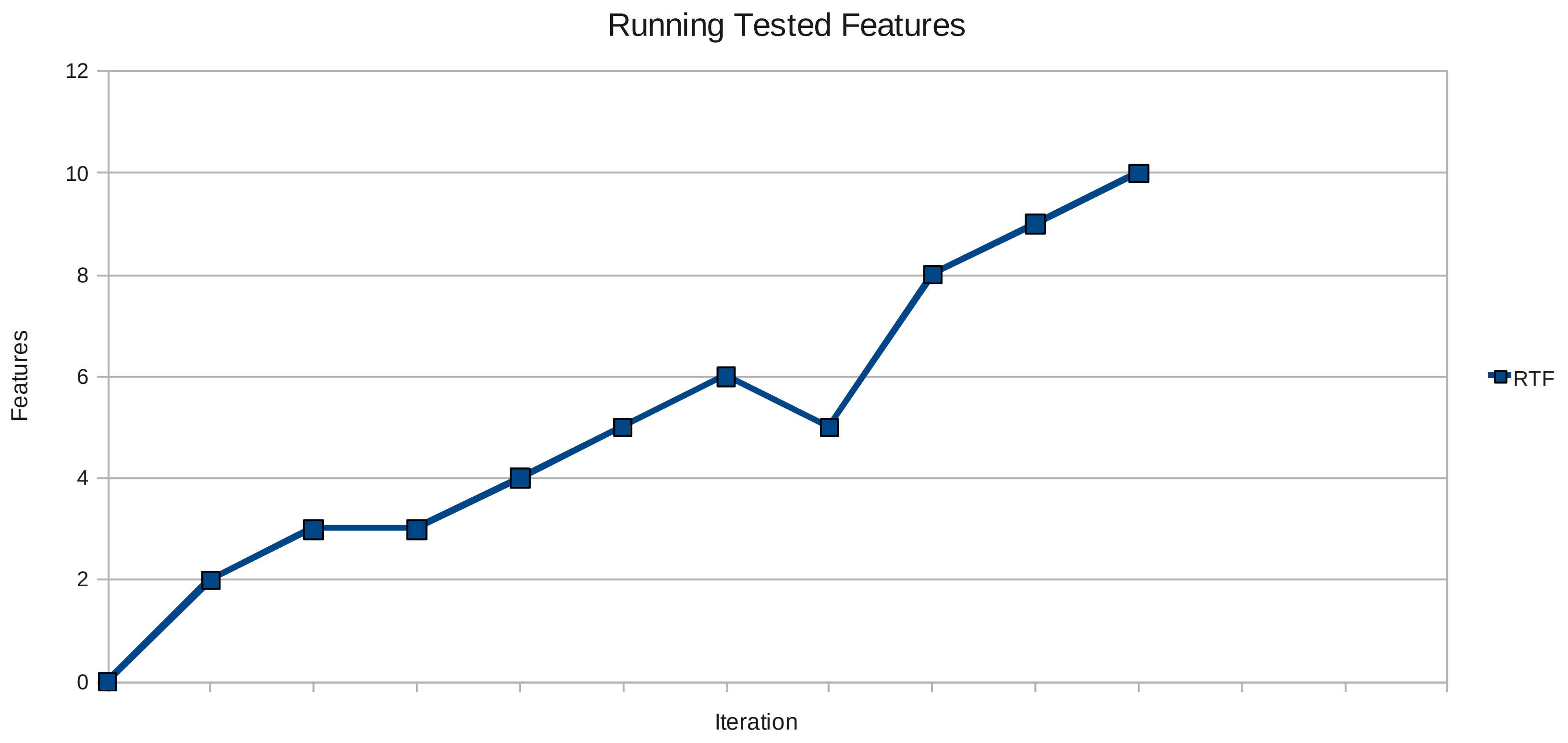
Practical Case Study

- Prepare the plan and scenario for Agile case study
- Sprint 2

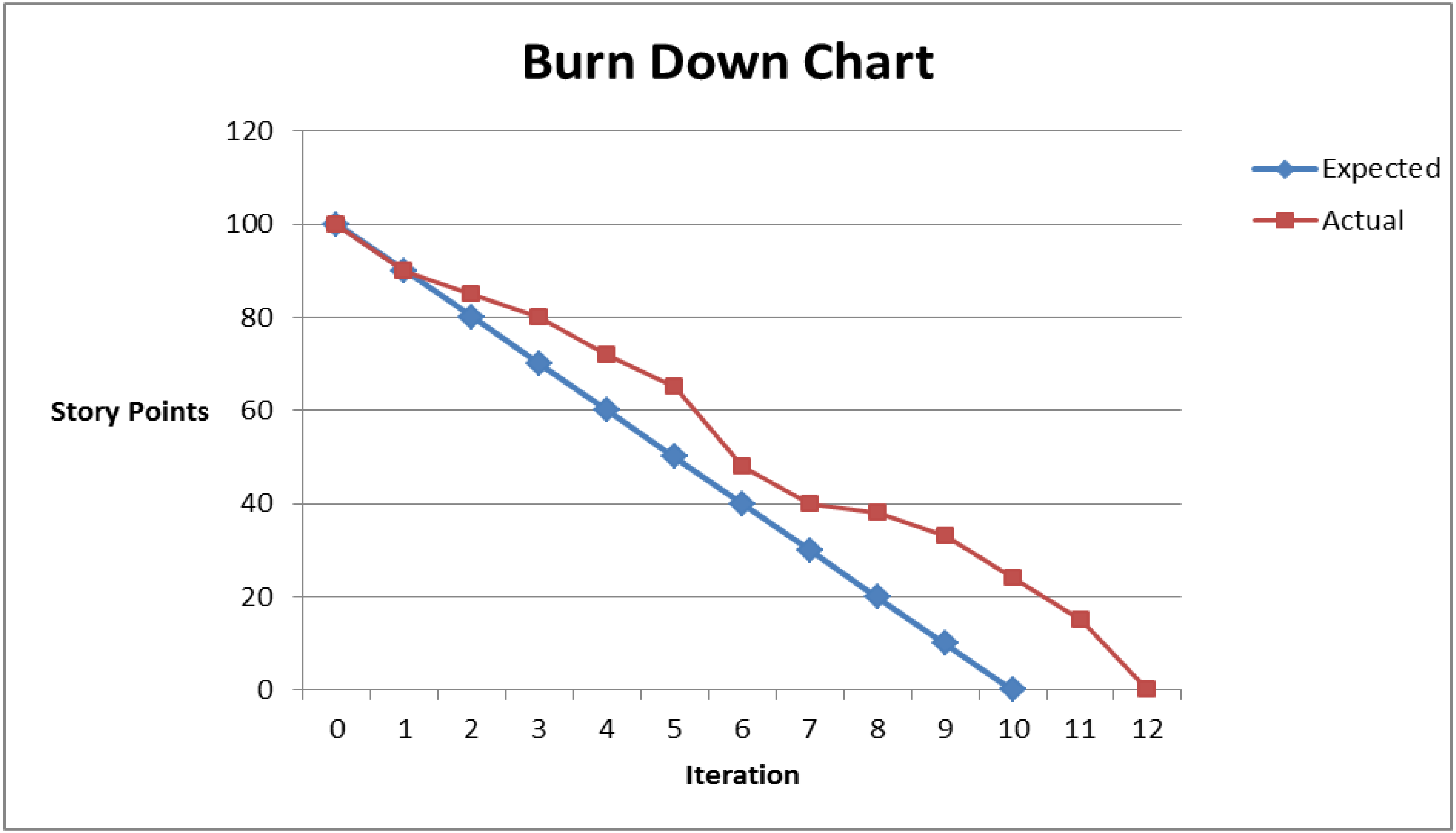
Practical Case Study

- Prepare the plan and charters for Agile case study
- Sprint 3

Running Tested Features



Burn Down Chart



Velocity

Velocity is...

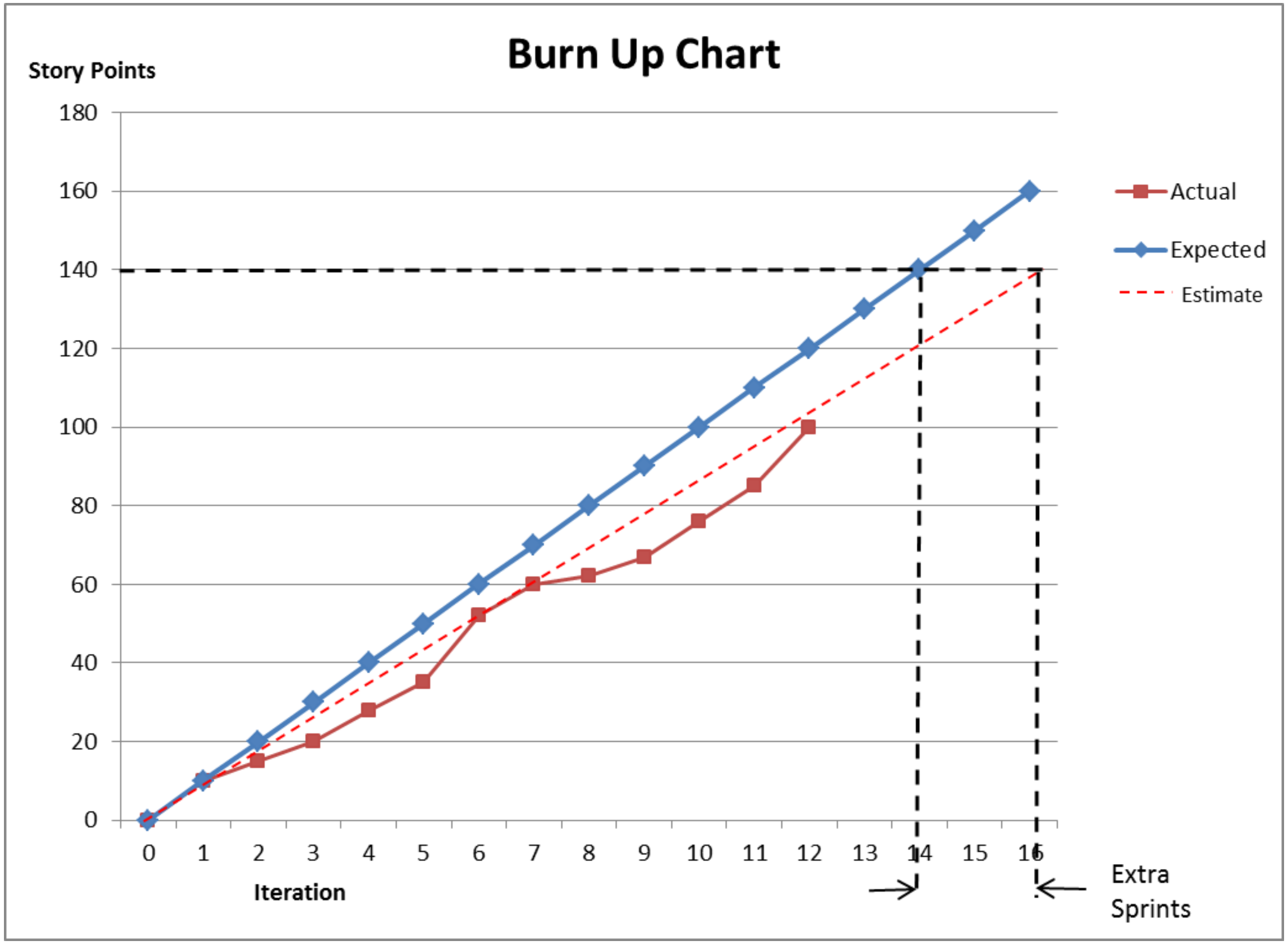
An empirical observation of the team's capacity to complete work per iteration.

...and not...

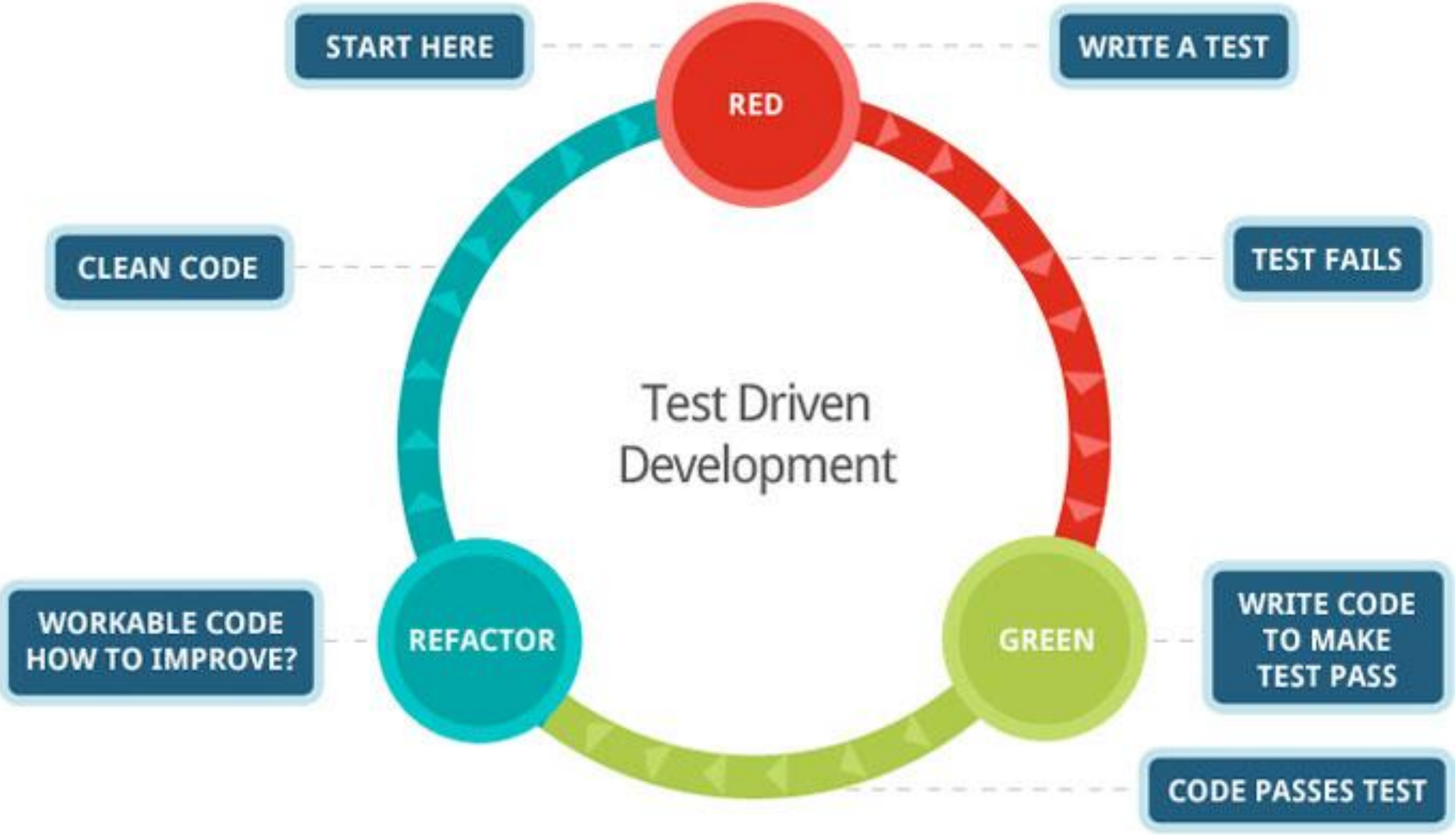
an estimate

a target to aim for

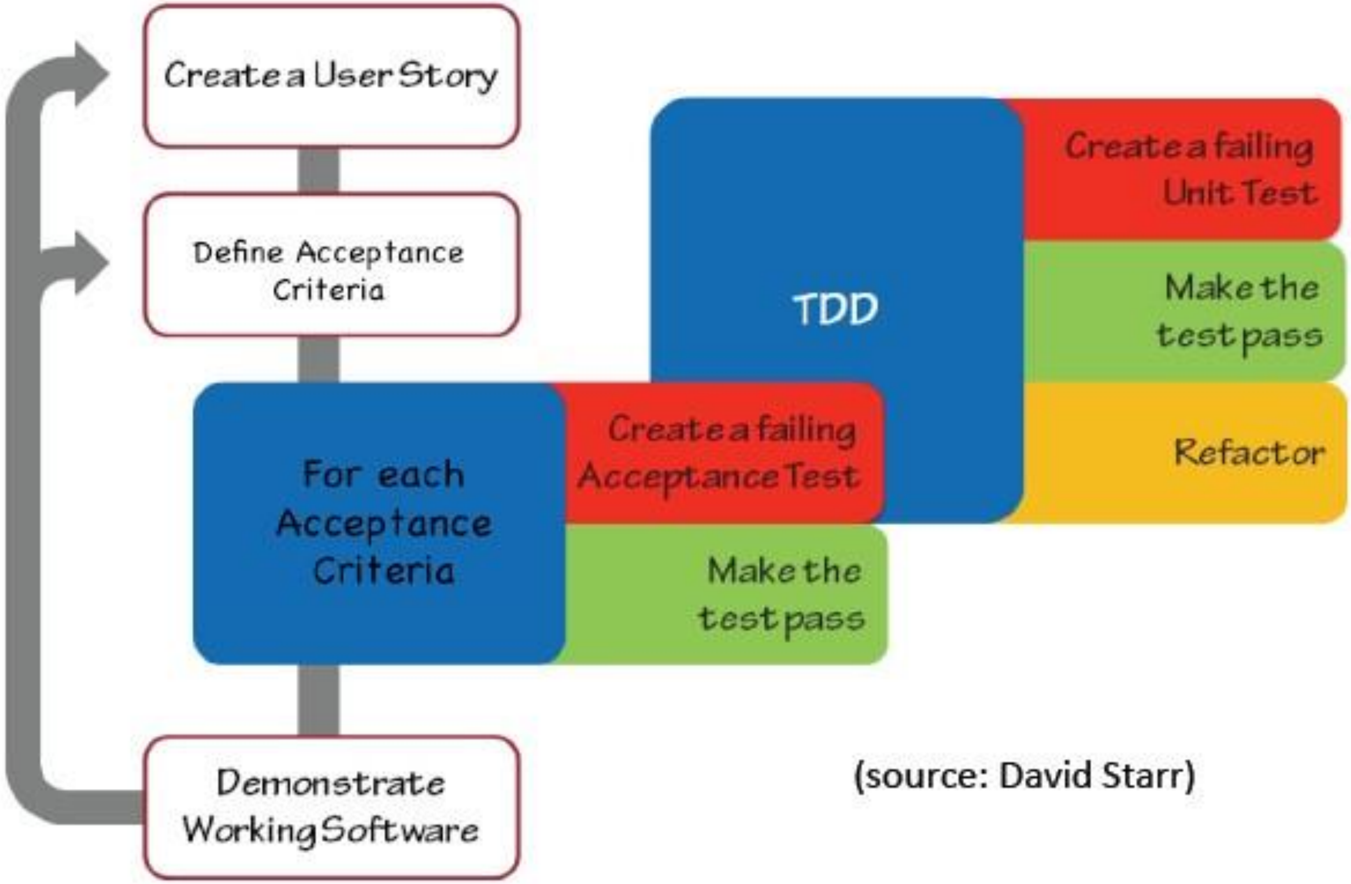
What Is Happening Here?



4.3 TEST FIRST


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven Development
WORKABLE CODE HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE TO MAKE TEST PASS
CODE PASSES TEST

ATDD



Create a User Story
Define Acceptance Criteria
For each Acceptance Criteria
Create a failing Acceptance Test
Make the test pass
TDD
Create a failing Unit Test
Make the test pass
Refactor
Demonstrate Working Software

(source: David Starr)

CASE STUDY ON TEST FIRST

What is Agile Testing

Quality is everyone's responsibility. Period.

Tester's Responsibilities:

Integral part of the team

Drive development with tests

Work with customers to define acceptance tests for each story / feature

Provide continuous feedback to the team

Provide constructive skepticism

Test each story as it is complete

Keep track of the "Big Picture"

The Whole Team Approach:

What is NOT Agile Testing

Testing is NOT a phase. Period.

Testing one or more iteration behind is NOT Agile Testing

Testing at the end of sprint is NOT Agile Testing

Writing too many test cases upfront and updating them just before every sprint is NOT Agile Testing

Using Automation tools to speed up testing is NOT Agile Testing

Not planning for automation is NOT agile testing

Using Agile Tools and Artifacts in Traditional Testing Models is NOT Agile Testing

Agile Testers Role

- Get moving! Be proactive!
  - Don’t sit and wait for things to come to you
- Who does what testing?
  - Understand the “Whole Team” approach
- Collaboration is key
  - Customers/product owners/business experts
  - Developers
  - Other team members

Tester’s Role

Traditional Tester Role

- Separate Test Team
- Testing happens at the end of development
- Testers work alone
- Testers act as gatekeepers
- No or little contact with business
- Automation created after development

Agile Tester Role

- Part of an entire team
- Testing happens parallel to development
- Testers pair with BAs, Devs and other testers
- Testers highlight risk
- Direct contact with Business
- Automation created before and during development

Questions Backlog

Thank You

Our social Media presence

Website: http://www.agiletestingalliance.org

Twitter handle @AgileTAlliance

Facebook Page: https://www.facebook.com/AgileTestingAlliance

Youtube link: https://www.youtube.com/user/AgileTestingAlliance

LinkedIn profile: https://www.linkedin.com/company/agile-testing-alliance/

SlideShare: Learning and sharing Presentations and information
http://www.slideshare.net/AgileTestingAlliance/
http://www.slideshare.net/ATASlides/