Introduction to User Acceptance Testing

TSG

Dawson House, 5 Jewry Street, London EC3N 2EX

www.testing-solutions.com

rgillan@testing-solutions.com (admin)

TestCon MOSCOW 2017

tsg

v1.3 TCM

About TSG

Software Testing Services

- We deliver targeted solutions for testing and assurance aligned to the needs of your business. From policy and strategy definition to testing approach and fully managed services, TSG are a full life cycle partner

Training

- Our certificated and non-certificated training courses provide the range of technical and managerial skills to increase organisational capability and support continual career growth and progression

Recruitment

- We support organisations in their quest for experienced and certified staff, be they permanent or temporary to meet a specific project requirement or demand shortfall. TSG works to reduce risk, cost and time to hire

About TSG

Areas of expertise

- Strategic Consultancy and Risk
- Agile Consultancy and Training
- Test Management
- Functional and Non Functional Testing
- Test Automation
- ISO 29119 Consultancy
- Certified and Practical Testing Courses

Introduction

Steve Portch

- Principal Consultant at TSG
- Started in IT as a mainframe operator in 1968
- Spent 15 years as a programmer in Assembler and C
- Started in testing at IBM Warwick Lab in 1989
- Major UAT projects for Halifax, Barclays, Old Mutual, ABN Amro
- Test strategies for Euroclear, Fiserv, Hastings Direct, Deutsche Bank

My first computer…

ICL 1904 Mainframe

64Kb RAM
240Mb Disk
3GB offline tape storage

Aims and objectives

- Understand what UAT is and what it isn't
- Understand the importance of user acceptance test and how it fits within acceptance testing overall
- Understand how to set measurable acceptance criteria
- Know when to do user acceptance testing activities during the software development or maintenance lifecycle
- Have been introduced to testing standards and to testing techniques

Timetable

Time		Activity
09:00	09:30	Introductions and course objectives
09:30	09:45	What is testing and why is it important?
09:45	10:30	What is UAT and where does it fit in?
10:30	10:45	Break
10:45	11:45	UAT preparation – defining and documenting tests
11:45	12:30	Test environments and test data
12:30	13:15	Lunch
13:15	14:15	UAT planning and estimating
14:15	15:00	The test plan document
15:00	15:15	Break
15:15	16:00	UAT execution, defect management and reporting
16:00	16:30	UAT completion
16:30	17:00	Q&A and close

What is testing?

Why is testing important?

Most of us have had an experience with systems that don't work as expected. This can have a huge impact on a company, including:

- Financial loss – losing business is bad enough but in extreme cases there are financial penalties for non-compliance to regulations
- Increased costs – these tend to be the result of having to implement a manual workaround but can include staff not being able to work due to a fault or failure and the cost of fixing the issues after go-live
- Reputational loss – if the company is unable to provide service to their customers due to software problems then the customers will probably take their business elsewhere

It is important to test the system to ensure it is error free and that it supports the business that depends on it.

Exercise 1 - Identifying tests

Take a couple of minutes and think of what you would want to do if you we given this to test:

Introducing User Acceptance Testing

What is User Acceptance Testing?

- User Acceptance Testing is often treated as the poor relation of testing and is given very little thought until it's time to actually do it.
- Then there is a mad panic to get something in place to be able to get user sign-off.
- UAT is probably the most important test phase of all as it is where we confirm that the system is fit for purpose to the business stakeholders.
- It is essential that UAT is planned properly, it starts as early as possible, enough time is allowed to complete the testing and to fix any problems that arise.

User Acceptance Testing defined

- There are many definitions of User Acceptance Testing, including these taken from the Internet:

  - *The goal of User Acceptance Testing is to assess if the system can support day-to-day business and user processes and ensure the system is sufficient and correct for business usage.*
  - *UAT tests are created to verify the system's behaviour is consistent with the requirements. These tests will reveal defects within the system. The work associated with UAT begins after requirements are written and continues through the final stage of testing before the client/user accepts the new system.*

- The primary objective of UAT is there to demonstrate that you can run your business using the system – it is fit for purpose.

UAT isn't...

- Development contingency
- A repeat of system test
- Testing against the detailed requirements
- Something you do in a week so that the supplier can claim acceptance and get paid
- Something that happens at the end of development (Standard waterfall diagram)
- An opportunity to change the design
- …

I know that's what I asked for, but it's not what I wanted

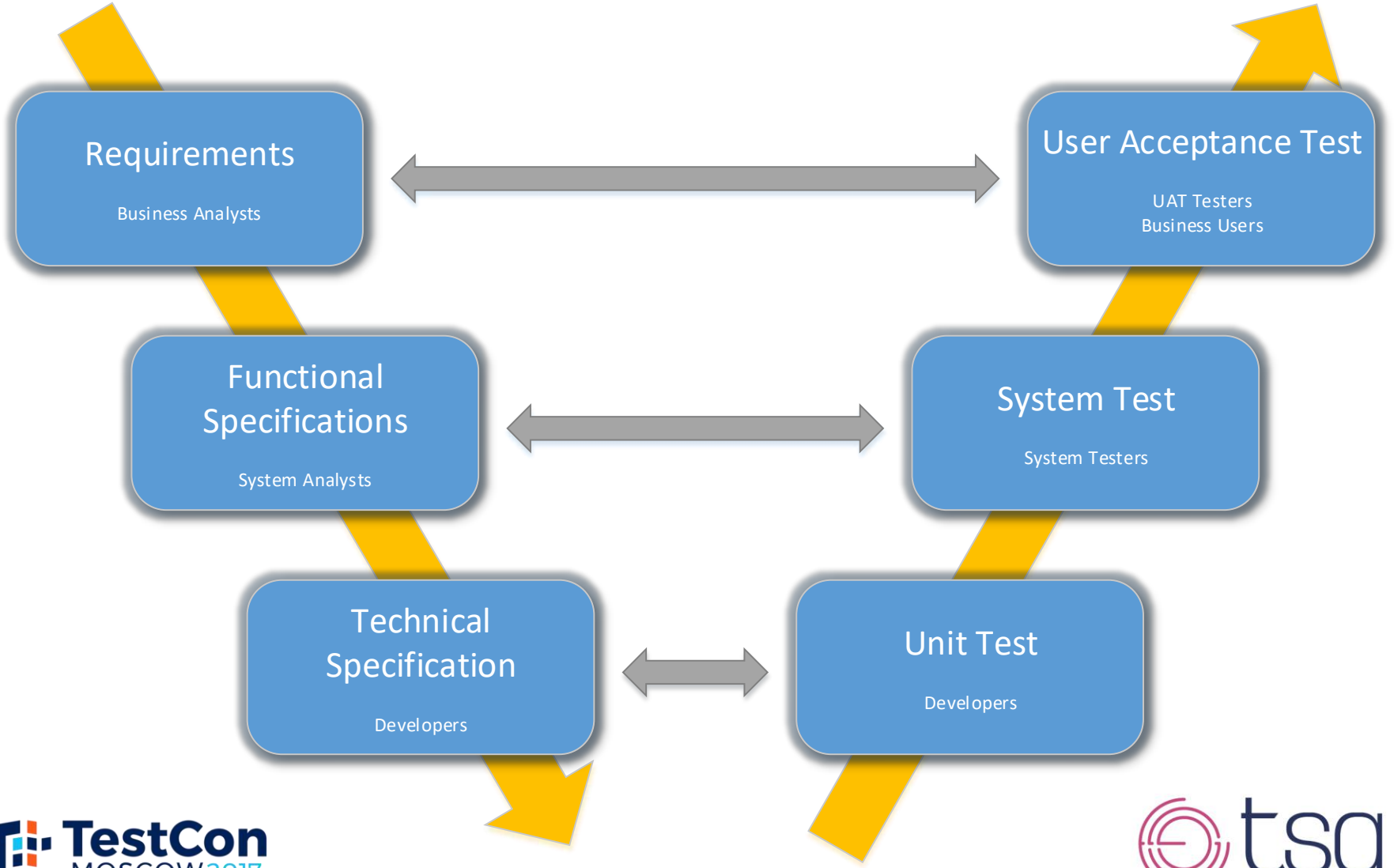
What makes a good UAT tester?

- A good UAT tester needs a number of different qualities:
  - **Background:** Good all-round understanding of the business and experience in business operations.
  - **Skills:** A good communicator at all levels, written and oral. Good understanding of IT systems
  - **Attitude:** An analytical mind, able to think around issues to find a solution and not easily put off.

- They need these because:
  - Understanding the business is key to being able to decide what should be tested and to what level.
  - Being able to provide concise, accurate and factual progress reports to the project team and accurate bug reports to development makes sure that the right information is provided for decision making.
  - Testing always finds problems and being able to find ways around them is an essential part of getting testing completed on time.

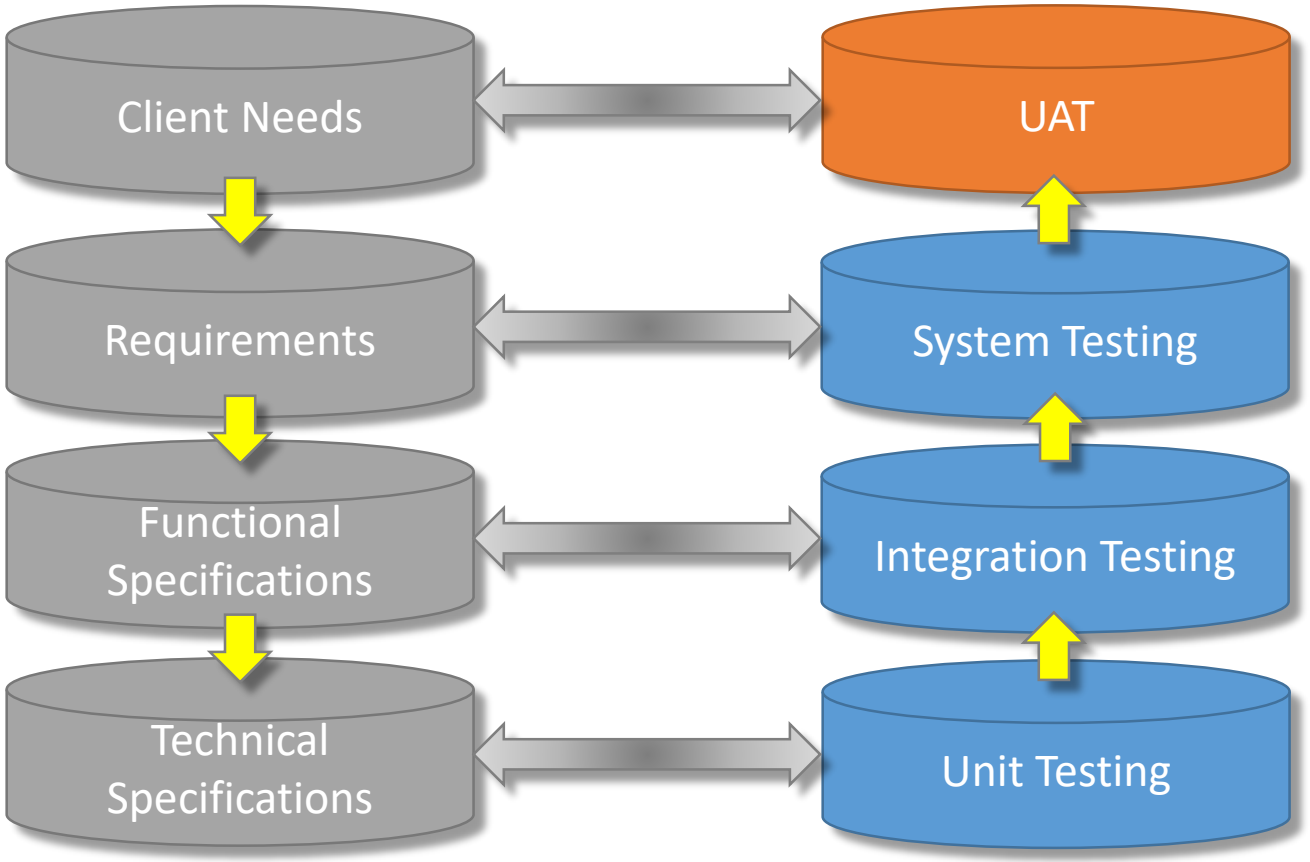
Where does UAT fit in?

UAT and the V-model



Software Development Lifecycle (SDLC)

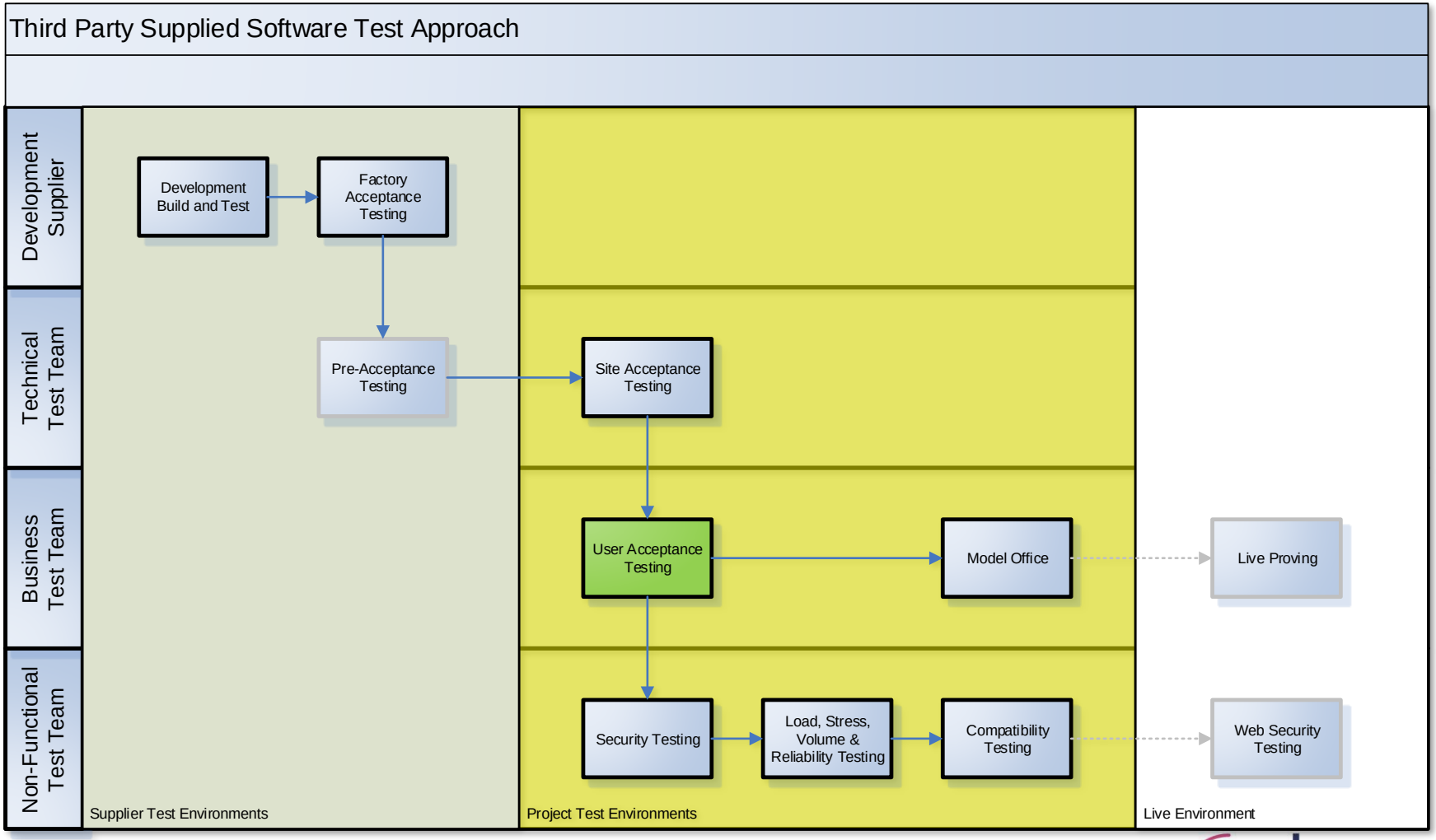
UAT and the V-model



Test phases

Test phase	Description
Unit Test	Performed by the developers to verify that the section of code that they are working performs to the detailed specification. Typically this will check that the controls work, data validation works as expected and data is input and output correctly. *Verify that the customer name field handles up to 30 characters.* *Verify that the system flags an error when any of the mandatory fields are empty.*
System Test	Performed by the test team to verify that the complete system performs to the specification and that the detailed requirements have been met. Typically this will check end to end processing completes with the correct results. *Verify that when returned goods are received that the stock balance is updated, the correct warehouse position is provided and a refund request sent to the finance system.*
User Acceptance	Once the full solution has been tested and delivered, the UAT team will validate that it meets the needs of the business. Typically this is performed by replicating the business activities on a close replica of the production system. *Validate that a customer service agent can authorise a goods return.* *Validate that a customer service supervisor can refund shipping charges to a customer.*
Non-Functional	In parallel with UAT, the test team will execute a further set of tests that verify the non-functional requirements such as browser compatibility, performance, security testing and may also engage a third-party specialist to perform website penetration testing. *Verify that the customer website works correctly using the latest version of Firefox.* *Verify that the system can handle 20,000 customers logged on concurrently.*

Where UAT fits in the SDLC



UAT lifecycle

- In the planning phase we identify what needs to be tested, estimate the time and effort needed and prepare the test plan.
- In the preparation phase, we document the test scenarios and make sure that everything is in place ready to run the tests.
- In the execution phase, we will run all of the planned tests twice which gives the development team time to fix any bugs we find in the first cycle and us to retest in the second.
- In the completion phase we report on what we found during testing to the project team as input to their go-live decision.

Exercise 2 – Classifying test cases

In your teams, go through the list of tests that you created in the previous exercise and decide which of them are User Acceptance tests.

When you've finished, work together to define up to five acceptance tests for a car based on the following user needs:

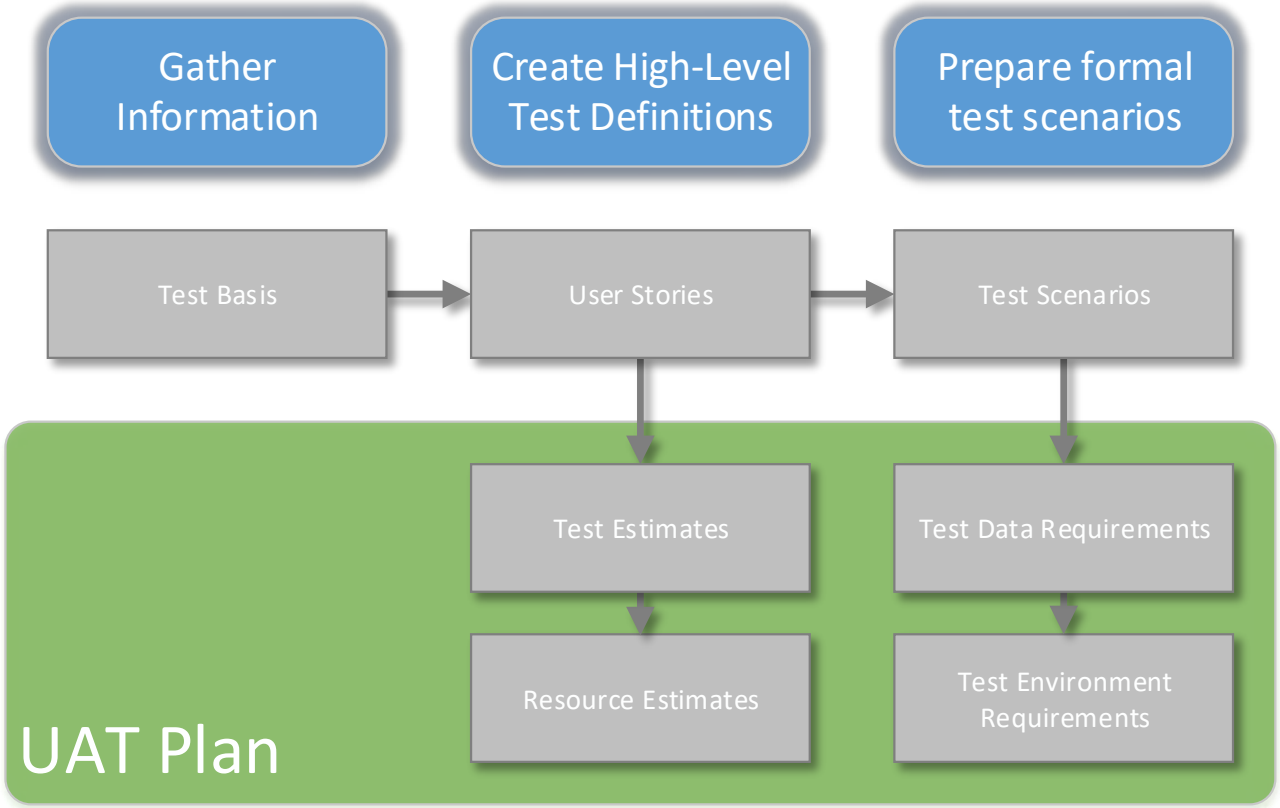
We are a family of four, two adults and two children under 5 years old, and we need a car that is suitable for local trips and our annual two-week holiday to Spain with the children's grandmother.

Break

UAT preparation

Test design process

- Test preparation is the cornerstone of UAT. It's where we get to understand the scope and create the tests we are going to need:



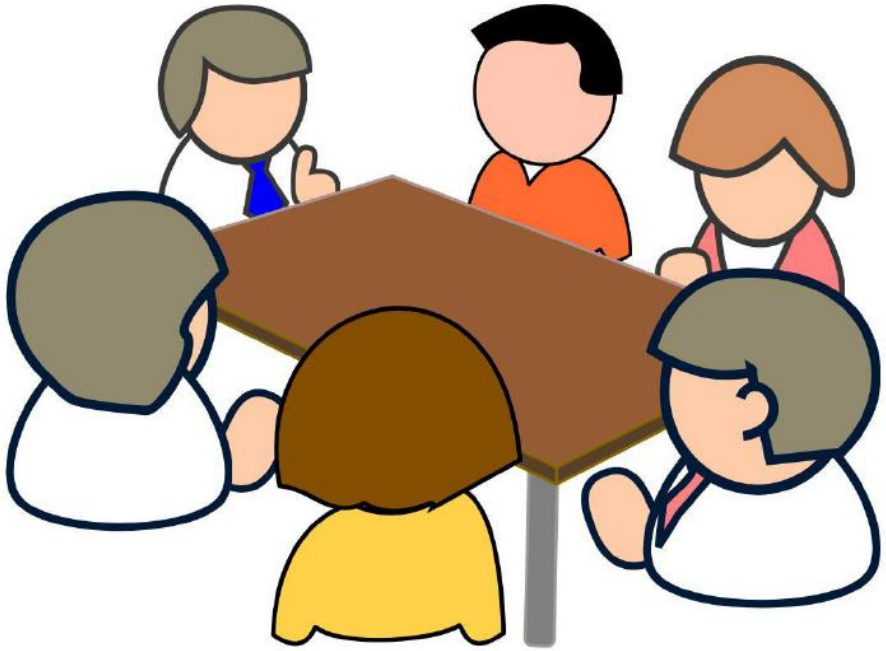
Designing UAT tests

- Good user acceptance tests:
  - Are based on the user roles and business processes that are needed to support the business
  - Are written in a way that a business user can easily understand
  - Only specify the data to be used where this is essential to the outcome of the test.
  - Are written so that a different user will execute the scenario slightly differently.
  - Clearly describe the expected results
  - Are verified by the business users before test execution starts

Test definition workshop

- The test definition workshop is one of the best ways to identify the scope and level of UAT testing that will be required. By talking to the actual users of the system it's much easier to find out:
  - What tasks are performed
  - How those tasks are performed
  - The variations of each task
  - Which tasks and variations are used most often
  - Which tasks and variations cause the most problems
- Although holding a workshop with several users from a department and members of the test team, a one-to-one meeting with the SMEs can be just as effective.
- The output from the workshop should be a process description for each task and list of the variations.
- This provides the inputs for estimating as well as test design.

Exercise 3 – Identifying UAT scenarios

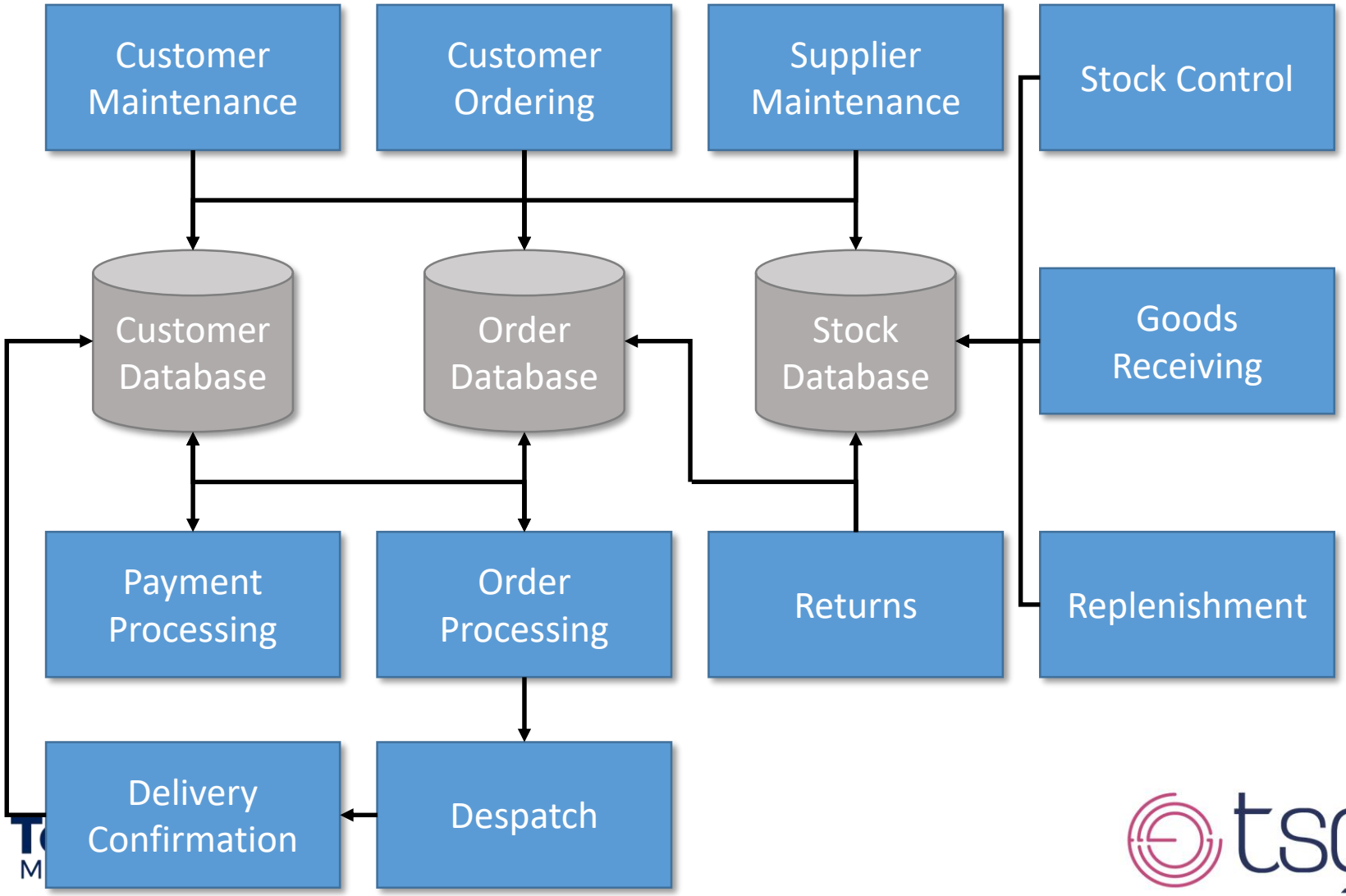


For this exercise, we are going to conduct a test development workshop using the business process for buying things from an e-Commerce site.

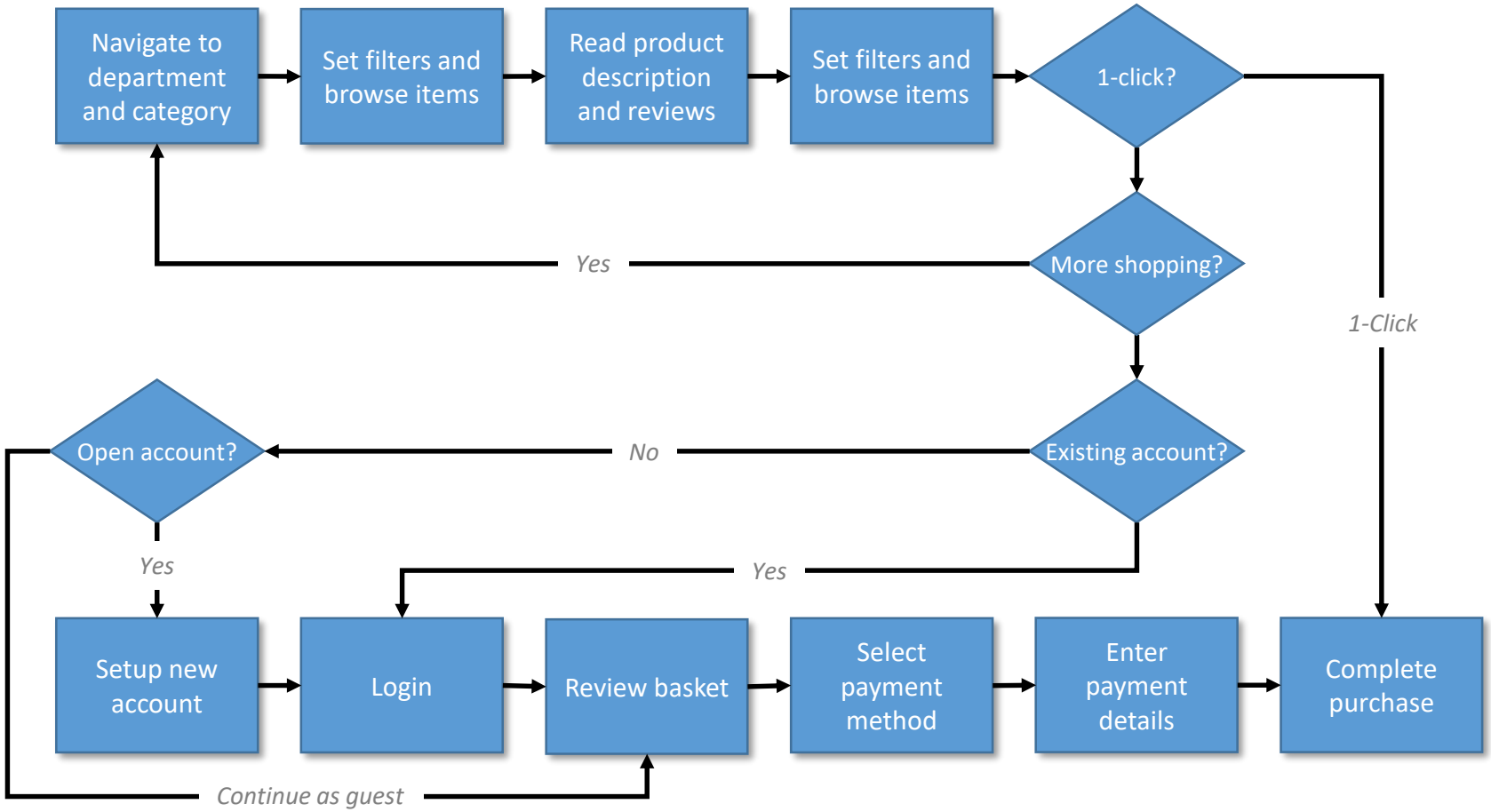
Your instructor will act as the facilitator and will ask some of you to act as the business SMEs.

Once we have completed the workshop and have mapped the process, working as a team, we'll have a set of one-line tests on the flipchart – we'll need them later…

e-Commerce System Schematic



eCommerce – Customer Ordering



e-Commerce – process flows

By following the process flows for customer ordering, we can create a list of tests that we could perform:

1. Buy a single item using 1-click
2. Buy a single item as a Guest
3. Buy a single item creating a new account and payment method
4. Buy a single item using an existing account and payment method
5. Buy multiple items using 1-click
6. Buy multiple items as a Guest
7. Buy multiple items creating a new account and payment method
8. Buy multiple items using an existing account and payment method

We could use this as the basis for writing our tests, but we can get a bit smarter and save ourselves some work…

e-Commerce – Initial tests

The testing can be simplified by considering what is being tested. By looking at the paths as a table we can see that we don't necessarily need to test everything in every combination to prove that it works from a user perspective.

Number of items	Purchase type
One	1-Click
One	Guest
One	New account
One	Existing account
Two	1-Click
Two	Guest
Two	New account
Two	Existing account
Three	…

But, we do need to assume that the development team have done their testing properly…

E-Commerce – Optimised tests

The objective of UAT is to check that the user can access the functionality they need to be able to complete a task, so we can be pragmatic in our approach. If we apply the rule of "test everything once" and we choose one of each purchase type and number of items, we have a table like this:

Number of items	Purchase type
One	1-Click
One	Guest
One	New account
One	Existing account
Two	1-Click
Two	Guest
Two	New account
Two	Existing account
Three	…

e-Commerce – Finalising tests

By doing this, we've been able to reduce the number of tests from nine to four but we can still confirm that the end users are able to buy items using each of the options:

Number of items	Purchase type
One	1-Click
One	New account
Two	Guest
Two	Existing account

Note that there's no test for three items as we've determined that we only need to test one item and more than one.

This uses a mechanical method to select the tests so we need to make sure that we haven't included a scenario that can't happen and that we have included the most likely ones by asking the business users to review and sign-off the list.

The test basis

- The test basis is the various sources of information that we will use to define our UAT tests. These may include:
  - Test definition workshops with the key business users (SMEs)
  - Training materials
  - Operational procedure manuals
  - Project initiation document
  - Business requirements document
- Wherever practical, asking users of the current system to explain what they do day-to-day is the best way of understanding the user scenarios that need to be tested
- You should also try and obtain information about which scenarios and variations are used most often as this is important for test prioritisation.

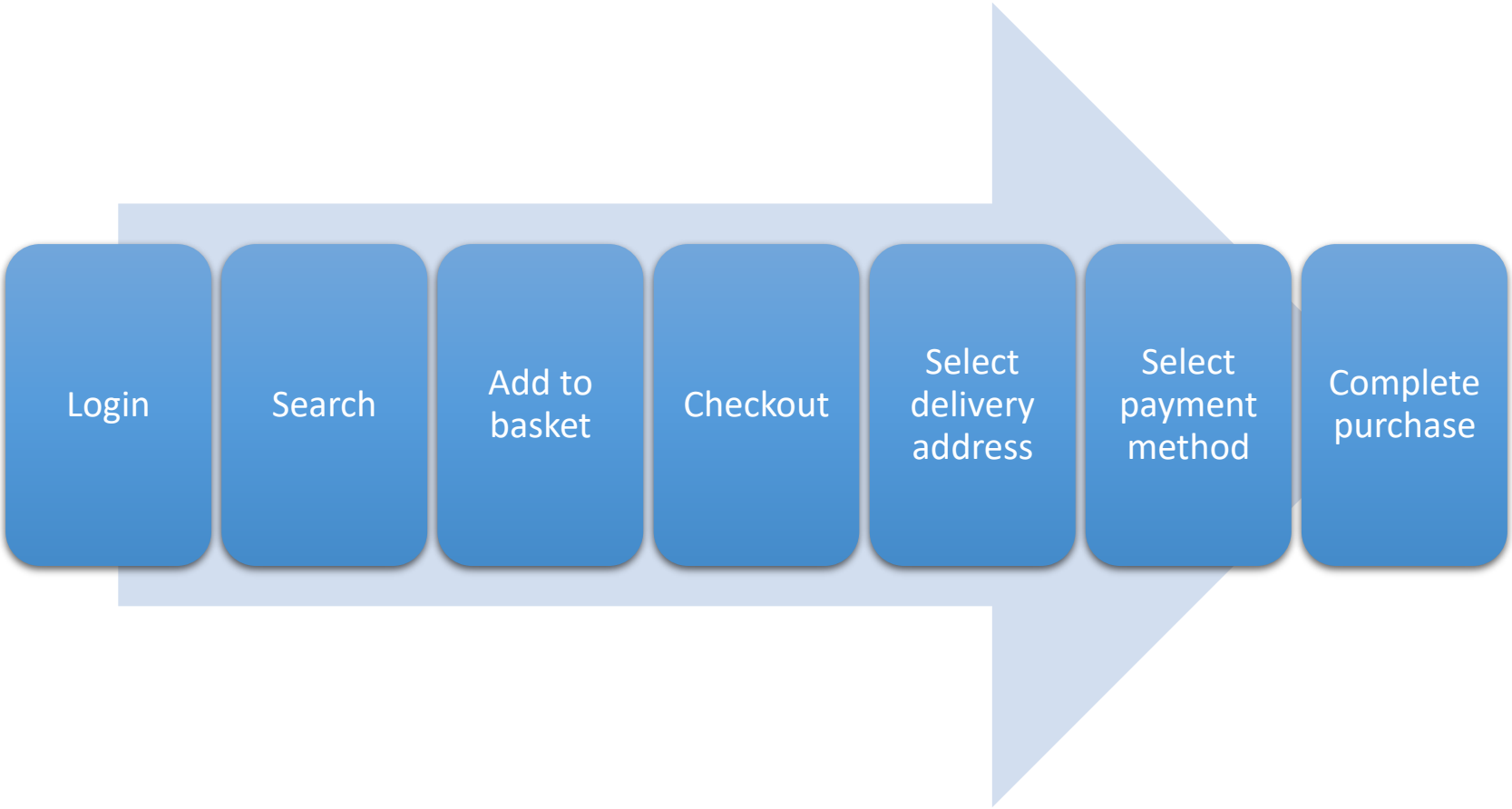
User stories

- One of the problem with testing is that it's quite easy to get dragged down into the detail, but UAT isn't about detail – it's about confirming that users are able to complete their day to day tasks using the system
- That's not to say we ignore the detail, but spelling mistakes on the data entry screen of an internal application aren't as important as not being able to amend a customer delivery address.
- User stories are a way of defining your tests that helps you keep the right level of detail. For example:
  - *As a customer service agent, I need to be able to amend the delivery address for a customer so that orders are sent to the right place.*
  - *I will accept this when it has been demonstrated that I can change the address on the system and the correct address is shown on the:*
    1. *Customer invoice*
    2. *Delivery note*
    3. *…*
- However, if we do come across bugs that should have been found in previous test phases, we will report them and expect them to be fixed.

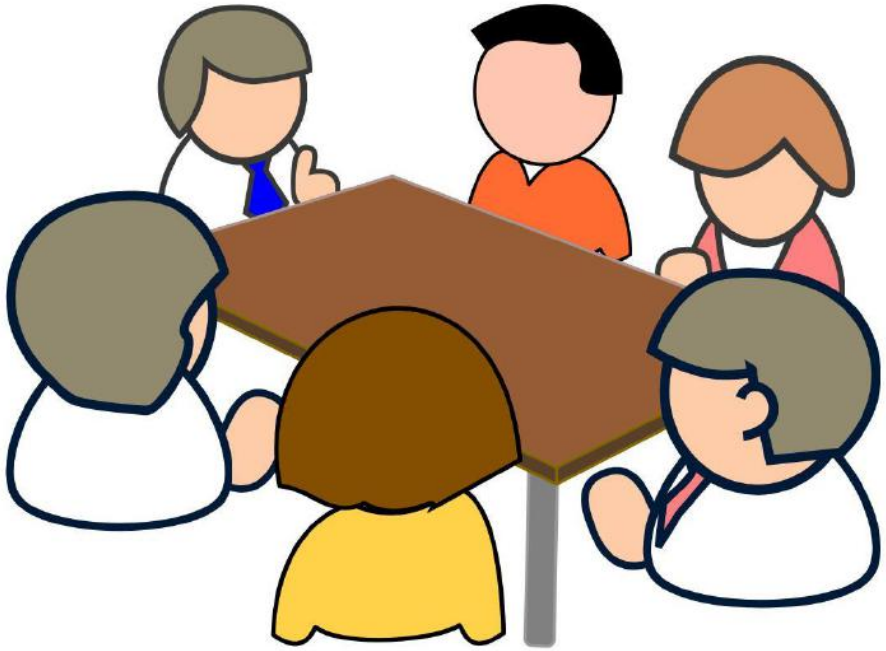
User stories

- User stories contain four parts:
  - The user role: *As a customer service agent*
  - The activity: *I need to be able to amend the delivery address for a customer*
  - The business value: *so that orders are sent to the right place*.
  - The acceptance criteria: *I will accept this when it has been demonstrated that I can change the address on the system and the correct address is shown on the:*
    1. *Customer invoice*
    2. *Delivery note*
    3. *…*

eCommerce purchase process



Exercise 4 – Writing user stories



Using the outputs from the test development workshop, we create user stories for each scenario that we identified…

We'll walk through a couple of examples, then you can try to do a different one yourselves…

e-Commerce – Test scenarios

Now we have our list of tests signed off, we need to document them in a format that the business users who will execute the tests will understand.

Unlike system tests which check the system to a high level of detail, UAT is intended to confirm that the users can complete their day to day tasks using it.

So, we aren't going to tell the users how to perform the test, just describe the task that they need to complete. Of course, the users need to have been trained on the new system before they start testing – which tests the training as well!

e-Commerce – Test scenarios

A good way of documenting the test scenarios is to write user stories. This ensures a consistent format and, if time runs really short, then we can get the users to just work from them.

Using the first scenario in our table as an example here's the user story and acceptance criteria:

As a customer I want to be able to purchase a single item without having to key in payment and delivery details each time so that I can place the order as quickly as possible. I will accept this when it has been demonstrated that I can place the order and I have received an order confirmation.

While this may seem to be an unnecessary step, it can make documenting the test scenarios much easier. In the next slide, the items in pink are taken directly from the user story…

Negative testing

- Generally, UAT is focussed on testing that users are able to complete their day to day activities.
- However, there are some occasions where UAT needs to confirm that the system prevents actions taking place, for example:
  - The system doesn't automatically update the delivery address when a change is made to the billing address unless they are the same.
  - When processing a refund, a user is not able to change any of the payment card details.
  - A banking system user is not able to make a discretionary payment to any of their own accounts.
- Detailed negative testing, like checking field lengths and invalid data entry, is performed in the earlier test stages and should not be part of UAT testing.

Exploratory testing

- As we've seen when defining a formal test, adding a controlled, random element into the tests is always a good idea.
- Exploratory testing takes this one step further without adding a significant extra cost.
- We simply allow some extra time for the tester to try a variation of the formal test they have just executed based on their recent experience.
- In our example, we expect a standard address such as 1 High Street, Anytown, AT14 1HS to be used.
- However, our tester may remember a valid, non-standard address that is just a PO Box and postcode e.g. PO Box 43, AT14 1PB and decide to try that out.

Test documentation

- Can be stored in a integrated test management tool e.g HP ALM, Enterprise Tester
- …or in Excel workbooks or Word documents
- Each test should contain:
  - Test ID
  - One line summary description
  - Priority
  - One paragraph scenario definition
  - Test priority
  - Pre-requisites and test data required
  - Test steps and expected results
  - Pass/Fail, Executed by and Date

Test scenarios

Description
Amend the delivery address for a customer

Scenario
As a customer service agent I need to be able to amend the delivery address for a customer so that orders are sent to the right place.

Step	Action	Expected Result	✓
1	Login to the system as a **customer service agent**	Login successful	
2	Find a customer	Customer record displayed	
3	**Amend the delivery address** and save	Updated customer record saved	
4	Create an order for the customer and save	Order is saved to the database	
5	Retrieve the order from the system	The amended delivery address is shown	
6	Print the **delivery note**	The amended delivery address is shown	
7	Print the **invoice**	The amended delivery address is shown	

Exercise 5 – Documenting UAT tests

Using the user stories that we created earlier, we now need to document our tests using a standard test case template…

We'll need to remember to define any pre-requisites or test data that we need to be identified at run-time

We'll work through one example then you can try it out on your own.

Your instructor will act as SME if you have any questions.

E-commerce - Test scenarios

Description
Purchase an item using one-click

Scenario
As a customer I want to be able to purchase a single item without having to key in payment and delivery details each time so that I can place the order as quickly as possible.

Step	Action	Expected Result	✓
1	Login to the system as a **customer**	Login successful	
2	Find an item to purchase	Item description is displayed	
3	Select **1-click** save	Item purchased and confirmation email received	
4	Retrieve the order from the system	The correct items are shown and invoice is available	

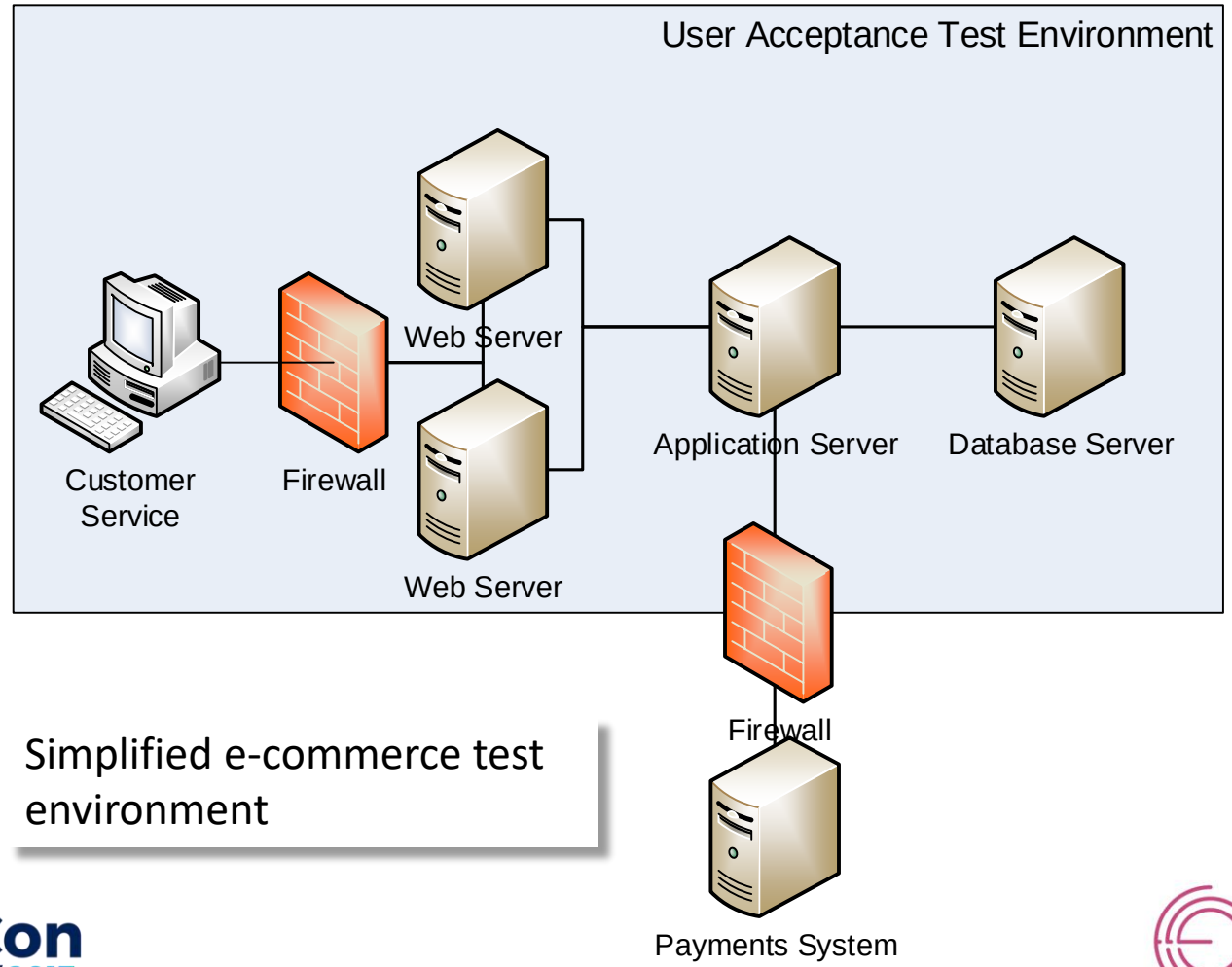
Note that there is very little detail in the scenario, we've only included the minimum necessary to complete the test. The user executing the test is asked to use any reasonable value for the test data e.g. the item to buy, which adds even more variation to the test.

Test environments and test data

Test environments

- The best test environment is a full replica of production with all of the interfaces to other systems in place.
- We then have a life-like environment for testing.
- The environment runs as if it is production including overnight batch processing.
- If possible, get the service operations team to look after the environment for you. If they are prepared to monitor it and auto-schedule any batch processing, then it makes life so much easier.
- After running the environment for a couple of weeks as if it were production without any operational issues, we can be reasonably certain that the system will work when it's deployed to live.

Test environments



Simplified e-commerce test environment

Alternative test environments

tsg

Test environments

- However, test environments can be very expensive, so we need to take a pragmatic approach…
- A scaled down replica of production is ideal as there won't be anything like the volume of traffic. For example:
  - Use smaller servers with fewer CPUs
  - Only have one server instead of a server farm
  - Less data storage (10% of production data is usually enough)
  - BUT, make sure that you have all of the production components
- Let the project team have your requirements as soon as possible and get the IT team to build and support it for you.
- Always get the IT team to set the environment up so that it runs like production including end of day/week/month batch processing.

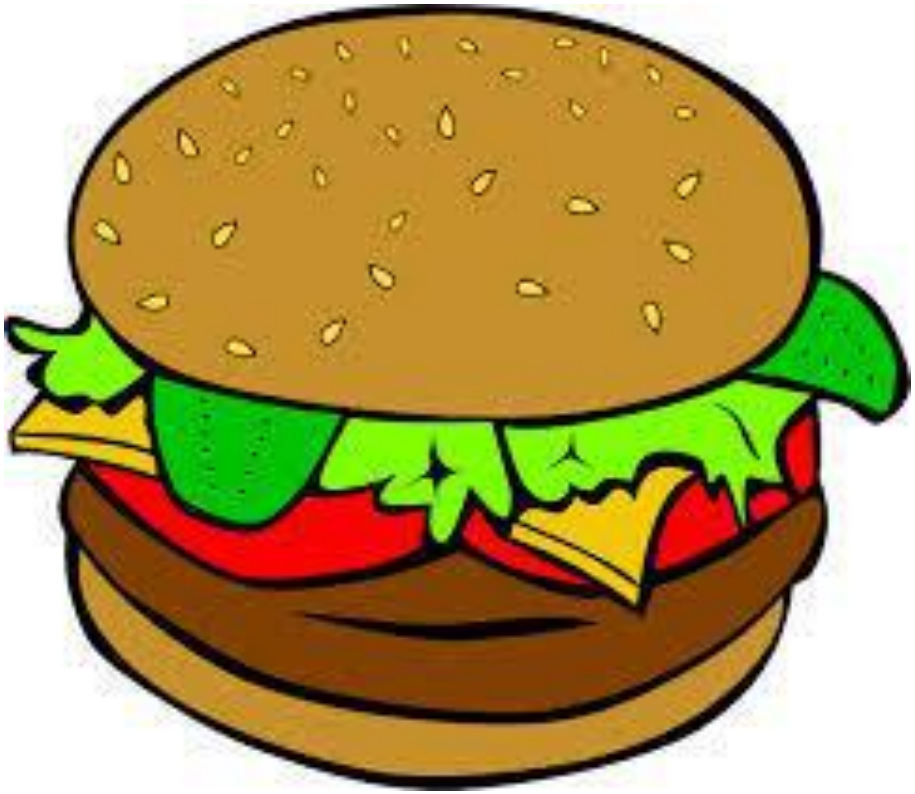
Test data

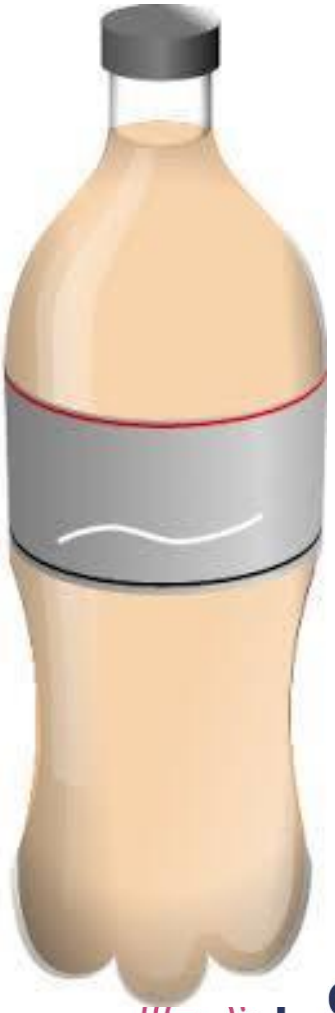
- Use a copy of production data wherever practical.
- Make sure any personal information has been obfuscated but be careful that this doesn't prevent you from identifying it.
- If data is to be migrated from an old system, use a copy of the migration output. Even if it's not 100% right it is still better than manufactured data.
- If you have to create your own, then use the system to do it – don't attempt to manipulate the underlying databases.
- If you need a lot of data, consider using an automated test tool to enter it into the system for you.

Defining data needed for a UAT test

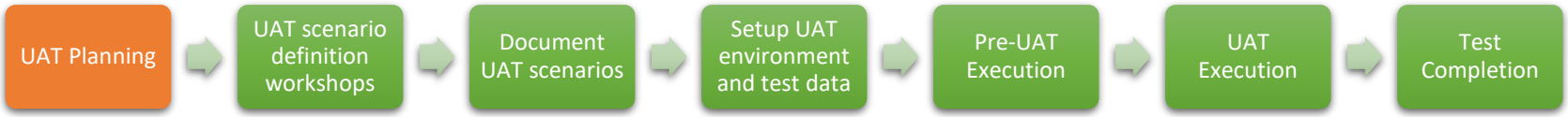
- Make sure that the test defines any pre-requisite data:
  - Existing customer with open, undelivered, orders.
  - Credit card account with a positive balance.
- Where the test requires data to be entered, make sure that it is realistic and different:
  - Always using Mr A Tester as the customer name makes it difficult to find the right one to be able check your results.
- Only define the parts of the data that are key to your test, let the person executing the test use reasonable values for everything else as this adds variation to your test.
  - Postcodes should be valid although the address shouldn't match
  - Define the SKU for testing out of stock, but not for customer purchasing – use the item description instead "a camera that is in stock".

Lunch





UAT planning



Test planning

- Now that we have identified the tests that we need to run as part of UAT, we need to:
  - Determine the most appropriate test approach
  - Estimate how much time and resource we will need
  - Define our UAT entry and exit criteria
  - Prepare the test plan document.
- Throughout this process, we will be making assumptions and we must keep a note of them as they are a key part of the test plan.
- An assumption documents something that we have based our planning on, but aren't able to confirm at the moment.
- For example, *It is assumed that the software supplier will have performed system testing in line with industry standards prior to the start of UAT.*

Test approach

- A typical UAT approach is to plan to execute all of the tests twice. This allows time for the development team to fix any problems and us to refresh the test environments before retesting. We usually allow 3-4 weeks a cycle.
- If you expect there to be lots of issues, for example it's an all-new system, then you could decide on three or four cycles, executing as much of the testing as you can in each.
- If you have lots of tests and limited time, you could decide to have three cycles but only execute each test twice.
- A more usual approach to the typical scenario of too many tests and not enough time is to have two cycles, but use a risk-based testing approach to decide which tests are going to be executed.

What is risk based testing?

- Risk-based testing is often assumed to mean that the project team are going to take a risk by not doing much testing!
- What it actually means is that available testing effort is concentrated on the areas where it is considered to be most needed based on a risk assessment.
- Tests are prioritised by assessing:
  - How critical this component is to the business operation
  - The probability that this component will fail
  - The complexity of the activity or underlying technology
- Assessing the risk in the context of UAT usually concentrates on the first statement, though the others need to be taken into consideration.

Risk based testing

- Risk-based testing (RBT) is an approach that prioritises the testing to be performed in a test phase based on the principles of:
  - Executing the tests in areas that have changed first
  - Executing the tests with the highest user impact first
  - Executing the most often used scenarios first
  - Always covering each business function with at least one test case
  - Varying the depth of testing (number of tests in each business area) according to the level of risk – low risk = fewer tests
  - Being prepared to adjust the plan if something unexpected happens
- If you use a risk-based approach, make sure that you get the business users to help set the priority and get written approval of your plan from the project manager.
- It's the project manager's job to accept risk, not yours.

Exercise 6 – Risk-based testing

As a team, we are going to use the risk-based testing techniques to prioritise the UAT test cases that we created in the previous exercise as High, Medium and Low.

We'll agree which ones we would want to execute if you only had time to complete some of them… this is how we apply risk-based testing to UAT.

Test estimating

- There are many different ways in which the effort needed for testing can be estimated including:
  - A wild guess
  - Previous experience
  - Simple formula
  - Complex formula
- Always allow time for defect retests (adding 25% is reasonable)
- Be realistic. We generally tend to underestimate the time needed to complete a task, so make allowances:
  - You could add a blanket 10%, but project managers don't like this
  - A better approach is to round the time required for a task up to the nearest 15 minutes
- Make sure you include time for management tasks and for supporting the users doing the testing during execution.

Simple formula based estimate

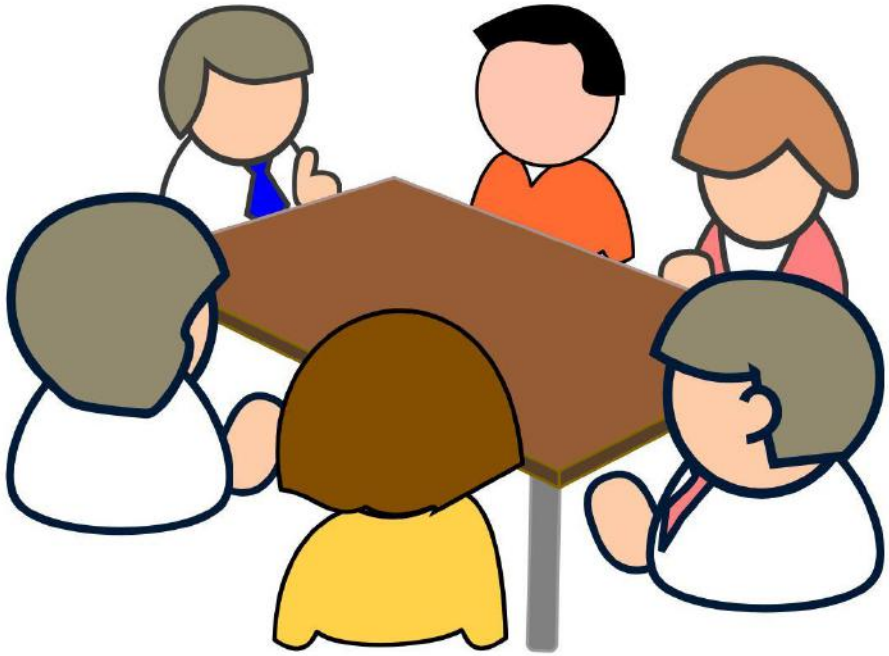
- The business specification shows that there are 50 functional requirements, we can use this as the basis for our estimates.
- From reading a sample of the requirements, we estimate that there will be between 7 and 15 tests per requirement.
- Taking an average of 11 tests per requirement gives us a total of 550 tests to be written and executed.
- Assuming it takes an average of one hour to design, document, review and sign-off a test and a further 30 minutes to execute it we can calculate the effort required:

Tests	Preparation		Execution		Total
550	Per test	Total	Per test	Total	825 hrs
	1.00 hrs.	550 hrs.	0.50 hrs.	275 hrs.	

Effort vs Duration

- Now we know how long we have for testing and the amount of effort we need to complete the work, we can quickly calculate the number of resources we need. Using the previous example:
  - 275 hours for execution at 7 hours per day = 53.5 days effort
  - Add 25% for retesting = 67 days effort
  - Add 20% for management and reporting = 80 days effort
  - We have 20 days to do the testing which means we need four people to do the work.
- The important thing is to have a sound estimating basis like this so you can demonstrate how you arrived at the numbers
- You may be challenged on your assumptions, but not on the calculation.

Exercise 7 – Test estimating



As a team, we are now going to decide on how much effort we think we will need for each of the following test activities:

1. Identifying the tests that will be needed
2. Documenting a test scenario
3. Writing a test plan
4. Executing a test case
5. Reporting a defect
6. Writing an exit report

Entry and exit criteria

- The entry and exit criteria describe what we must have in place to be able to start testing and what we need to have done in order to finish UAT:

A typical set of test exit criteria might look like this:

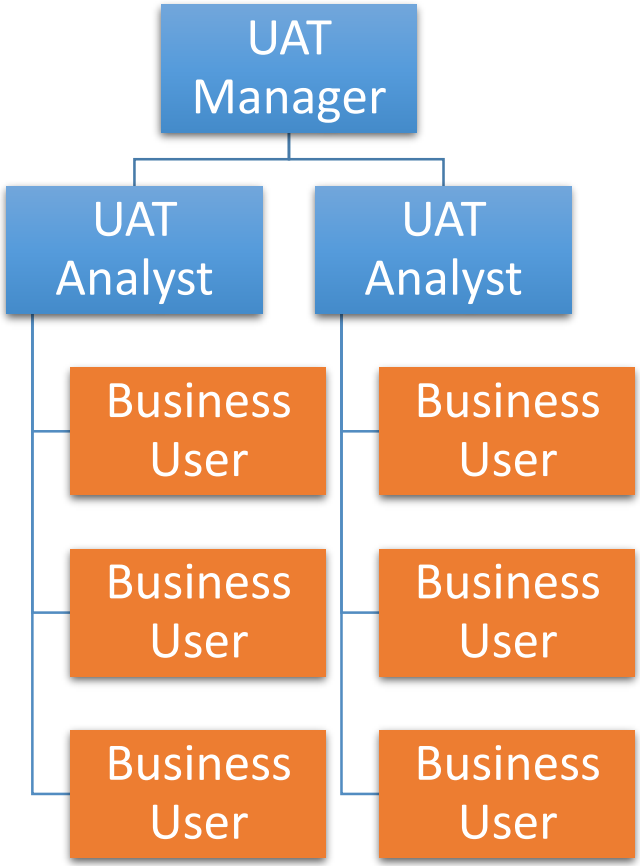
- Test completion report prepared
- All tests completed and signed-off
- No severity 1 & 2 defects open
- A maximum of 10 severity 3 defects open

Are these better?

- Test completion report prepared
- All key business functionality demonstrated to be working as expected
- No priority 1 defects open
- Action plan in place to resolve all open priority 2 defects by go-live

The UAT team

- The diagram shows how the UAT team for a project implementing a third-party developed software package might look.
- The UAT Manager is responsible for creating the test approach, test planning and resolving issues.
- The UAT analysts are trained testers who design and prepare the tests and support the business users during test execution.
- The business users help define the test scenarios and perform the testing once the system is ready to test.



The UAT Team

- Wherever possible, it's always best to use operational users to define and execute UAT tests as they understand how the business works much better than anyone else.

- But users don't generally have experience of testing so, as testing professionals, we need to make things as easy as possible for them by:
  - Writing the test scenarios as business operations not 'test-speak'
  - Scheduling testing around their operational commitments
  - Making sure that the system and data is available
  - Providing assistance when they have problems
  - Verifying the problems they find and reporting defects
  - Recording the test results
  - Retesting defect fixes

The test plan

- Is essentially the contract between the test team and the project manager and stakeholders and should clearly state:
  - What is to be done and, more importantly, what isn't
  - What needs to be in place for testing to start
  - How much it will cost, in terms of time and resource
- Implements the organisational test strategy, if there is one, which will contain:
  - The test approach to be taken for each test phase
  - Roles and responsibilities
  - Processes and procedures
- However, it may be necessary to deviate from the strategy for a particular project, these must be documented in the test plan.

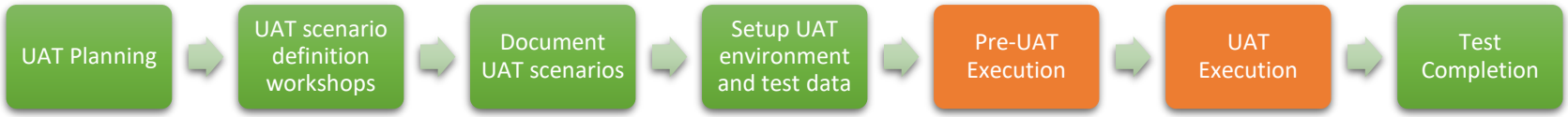
Content of a test plan

The Test Manager will prepare a UAT Plan which will contain:

- The approach to testing in this stage including the number of testing cycles planned.
- The testing that is to be performed in this stage.
- The test basis
- The test types that are in scope.
- Any deviations from the standards contained in the organisational strategy.
- The detailed entry and exit criteria.
- The testing resources that will be needed.
- The test environment and tools that will be needed.
- The test schedule.
- Test data requirements.
- Any risks, issues, assumptions and dependencies.
- A summary list of test scenarios to be executed.

Break

UAT execution

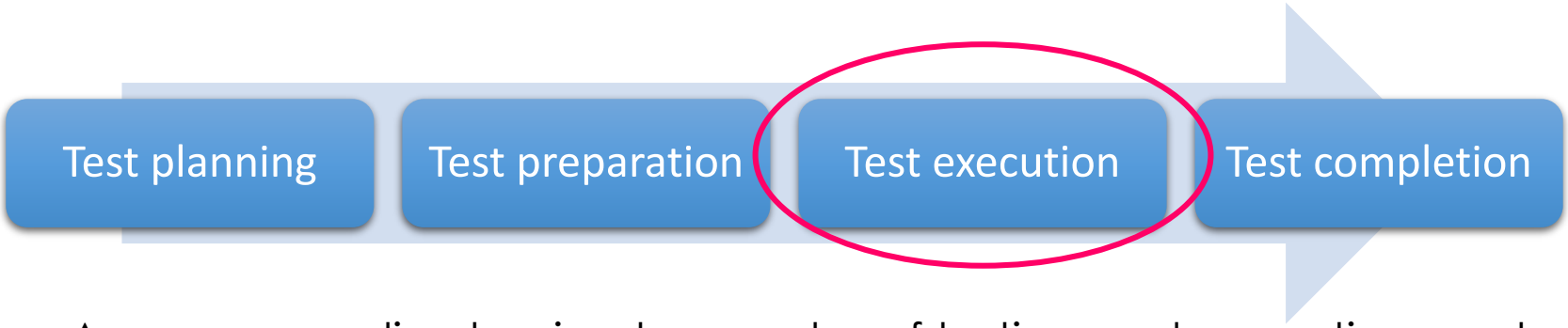


Test execution

- Now you have your test plan, test cases, test environment and test data and the new system has been delivered, it's time for the easy bit – running the tests.
- If the testers in the previous stages have done their job then all you need to do is to quickly run through your tests and pop off for a well-earned cup of tea.

- Oh, if it was only that easy…

Test execution

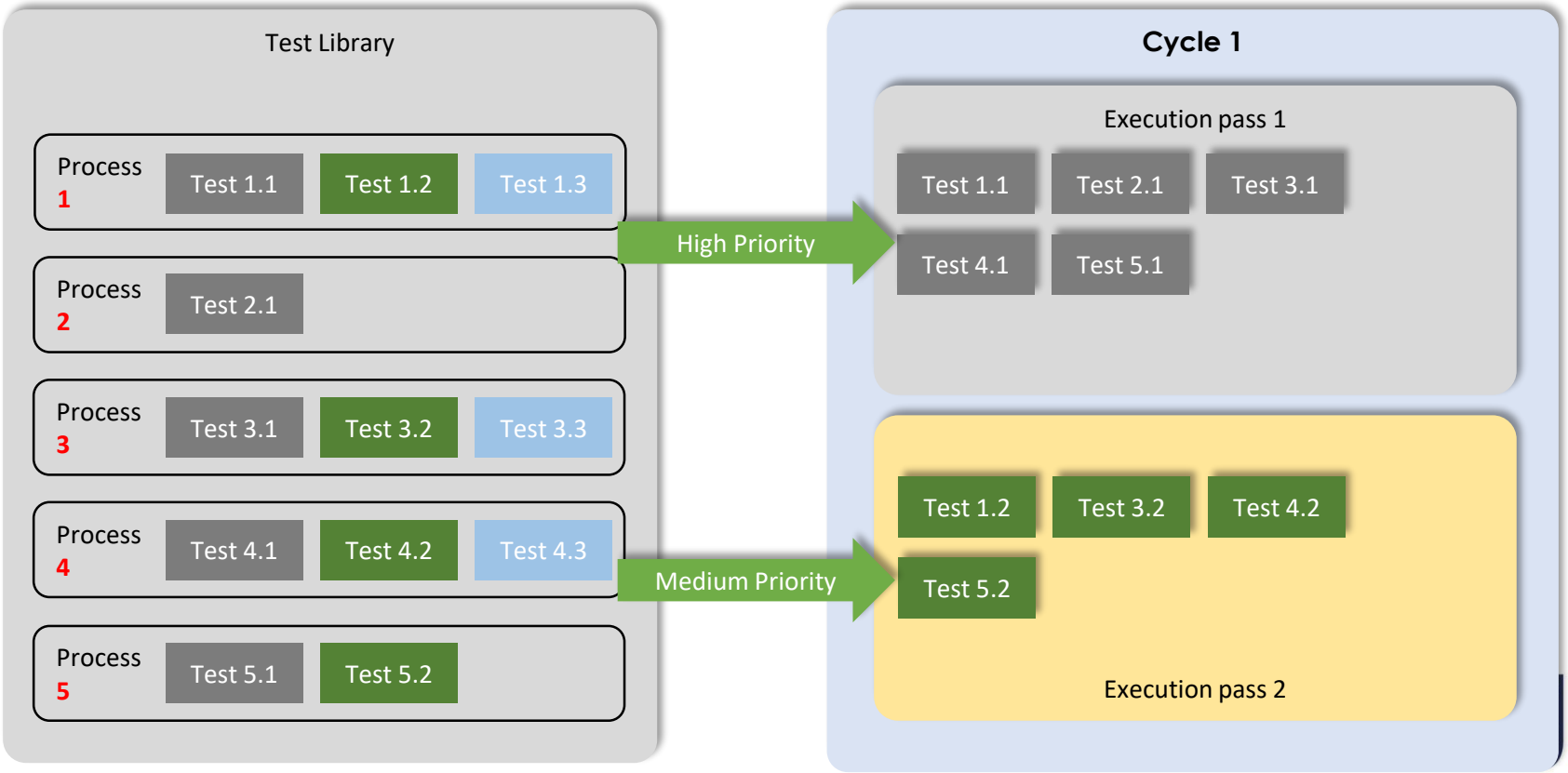


- As we saw earlier, having two cycles of testing and executing each of the tests twice is a fairly standard approach, but why?
  - It allows time for the developers to fix any problems, time to refresh the test data as we may have introduced errors and for us to have time for retesting defects.
  - If we find a critical defect with the last test we planned to execute, we have time for it to be fixed and retested during cycle 2.
  - We could run one cycle and just retest the bug fixes, but it's always possible that the fix for one problem causes another somewhere else, so having a second cycle gives us a good chance of finding it.
- Remember that the objective of UAT is to confirm that the system is fit for purpose and we need as many tests as possible to have passed.

Risk based test execution

When using a risk-based test approach, we still have two cycles, but each cycle is split into two passes and pass 2 won't start until we have assessed the results.

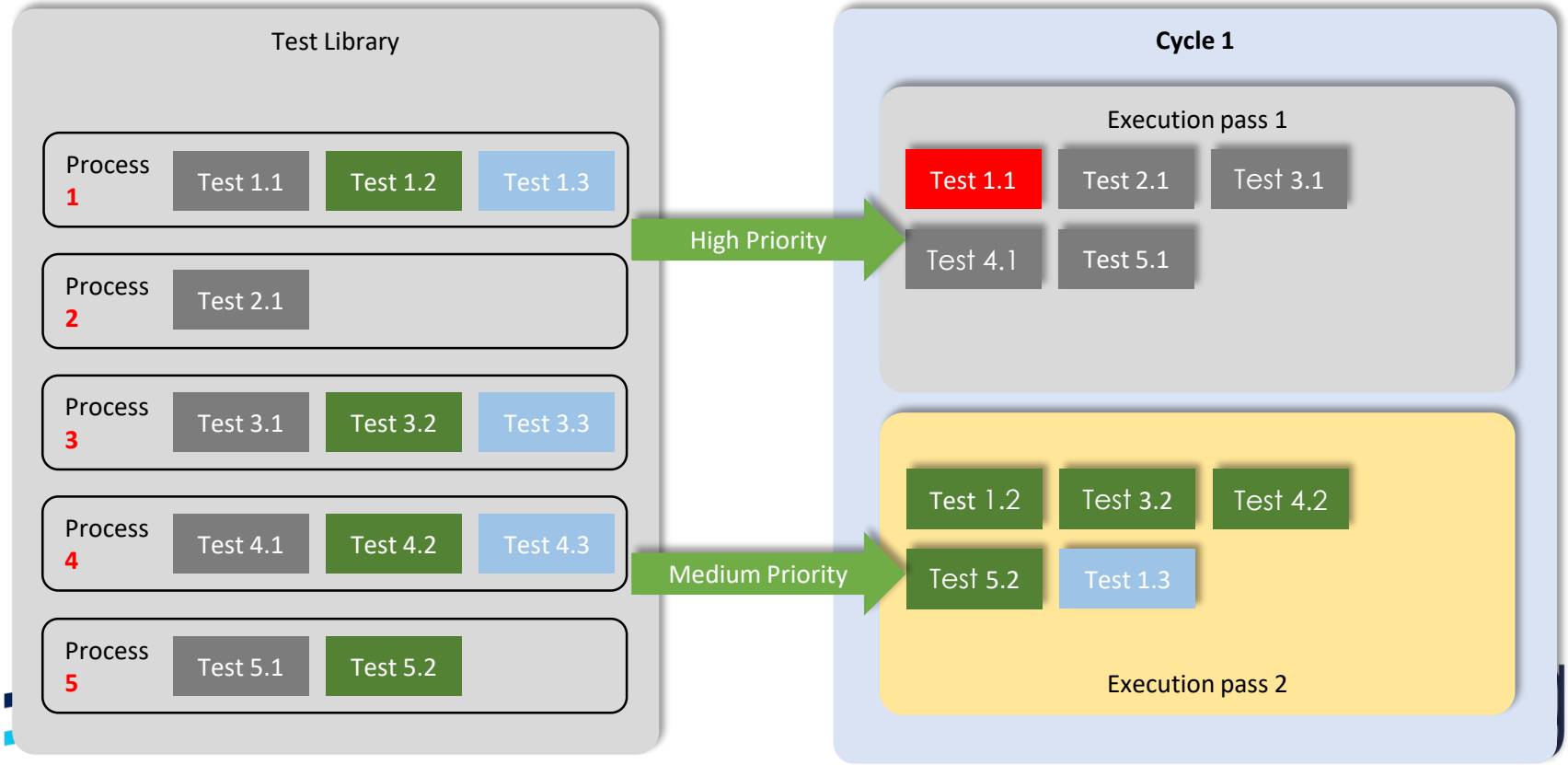
We are going to execute the high priority tests in pass 1, the medium priority tests in pass 2, but, at the moment, we aren't planning to run the low priority ones.



Risk based test execution

Here are the results of our first pass of testing and unfortunately our high priority test for process 1 has failed unexpectedly, so we have added the low priority test to pass 2 so that we can see if the fault is confined to the failed scenario.

We may need to reassess and remove tests from cycle 2 to fit the schedule.



What can go wrong?

- Pretty much anything and everything!
- So it's important to be prepared and to know what to do when it does:
  1. Take a screen shot and make a note of any error messages
  2. Write down what you did to get to the error
  3. Clear the error and repeat the test to confirm that you can reproduce it.
  4. Check your test case to make sure you haven't got anything wrong e.g. if your test says enter an X and the field title says enter A or B, it will fail!
  5. Check that you are using the correct test data
- If you are happy that the problem can be reproduced and you haven't made a mistake in your test, then report a bug!

Reporting defects

- Defects (or bugs) are generally reported using a test tool such as ALM or JIRA. These tools allow defects to be tracked through to resolution and provide key quality metrics.
- To ensure that the developers are able to fix the problem quickly and first time, it's essential that the testers provide clear, concise and accurate defect reports:
  - Write a concise and accurate one-line summary of the problem
  - Include all of the steps you took to find the fault
  - Include the screen shots you took as evidence
  - Imagine that **you** will have to test the fix using only the information in the defect report - if you can't then neither will the developer
  - Make sure you assign the correct priority

Defect priority and severity

Defects are assigned a severity and priority. The severity is the impact of the issue if it were to be found in live service, the priority shows how quickly it needs to be resolved.

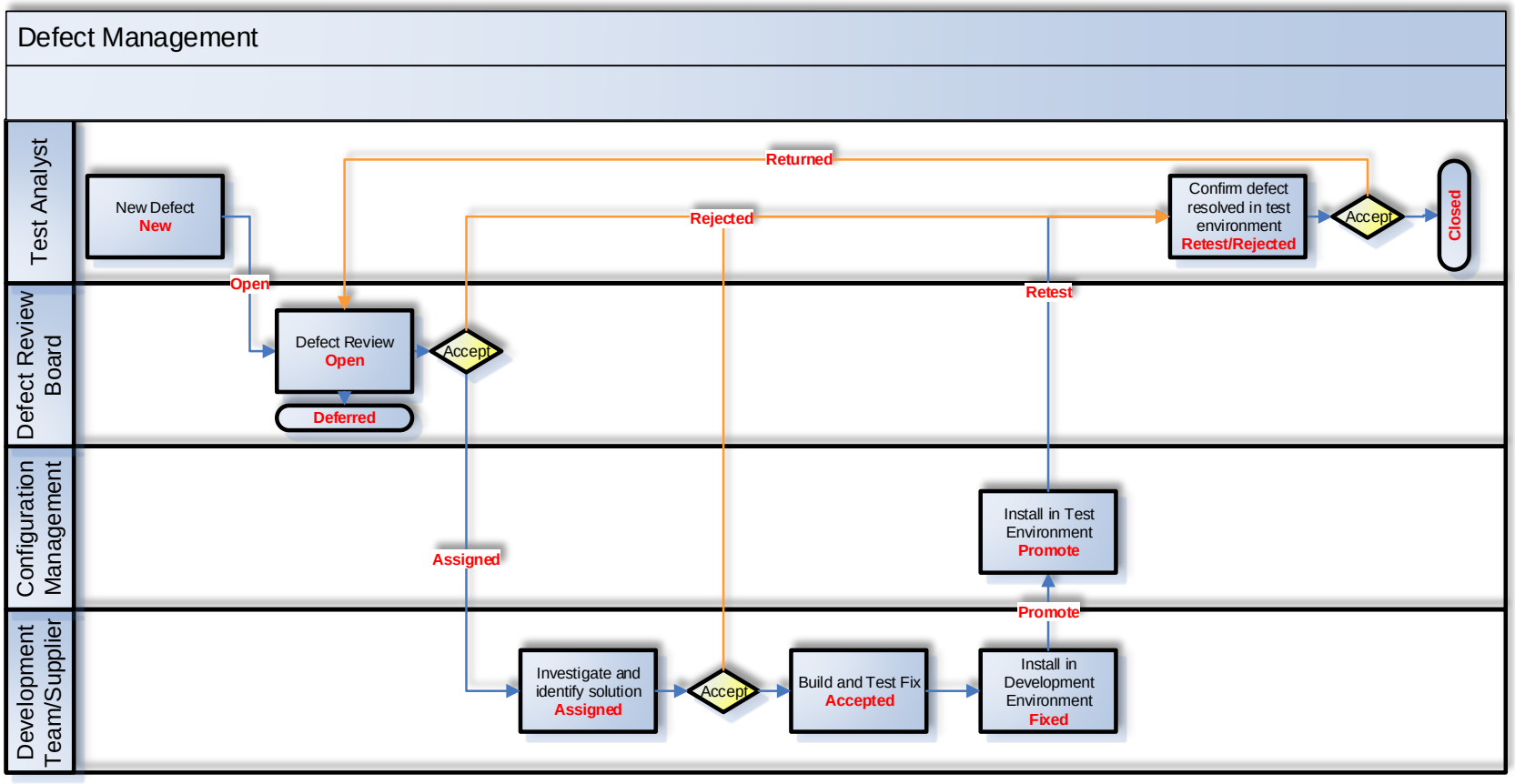
Severity	Definition
1 Severe	A significant part of the system is not working. There would be a breach of financial regulations or the company would be exposed to significant reputational risk if this were found in live. The system cannot be released to live without this being fixed.
2 Significant	Key functionality is missing or does not work. There would be a significant impact to business operations or the customer experience if this fault were to be found in live. The system should not be released to live without this being fixed.
3 Moderate	A part of the system functionality is not working to specification. The fault may impact business operations, but will not affect the customer. The system could be released to live with a suitable workaround.
4 Minor	There is a minor error in the functionality or a cosmetic issue. There is no impact to the business operation or customer experience.

Defect priority and severity

The priority shows how quickly we need the issue to be resolved. This tends to change on a daily basis, especially as the project deadline approaches.

Priority	Definition
1 Critical	Testing is unable to continue or is being seriously impacted. The business cannot operate until this defect being fixed.
2 Major	A significant amount of testing is being impacted. There would be a major impact to the business operation if this were not fixed.
3 Minor	There is a minor impact to testing. The business is impacted, but a workaround may be acceptable.
4 Trivial	Accepted that this is a fault, but there is no impact to testing. The business could operate without this defect being fixed.

Defect management process

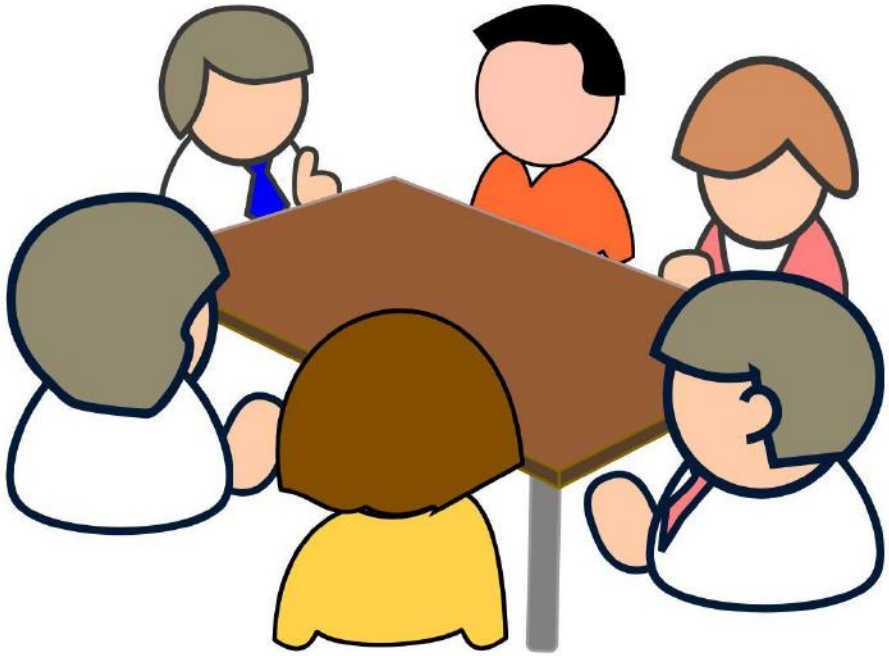


Defect resolution

- It's important to understand that not all defects will be fixed by the developers making a change to the software. This can be for a number of reasons, including:
  - It may be a change to the requirements
  - While it is definitely an error, it cannot occur in live service
  - The fix is extremely complicated and may destabilise the system
  - It is a low severity issue and can be deferred to a later release

- So, we tend to think of a defect being resolved, which can be:
  - The developer has fixed the issue, you have retested and found it OK
  - A manual workaround has been accepted by the business users
  - The project manager has confirmed that the defect does not need to be fixed in this release in writing.

Exercise 8 – Test execution



Individually, choose one of the tests from the set that we created earlier against the Amazon website, but please do not actually buy anything.

TSG will not pay for anything that you do buy!

Then, try executing the same test against a different eCommerce site – you should find that it works just as well.

Make a note of the actual results and any defects you find… we will discuss them when you have finished.

Test reporting

Reporting

- Daily report
  - Used to show progress (or the lack of it) to the project team
  - Information not data
  - Be factual
  - Use numbers to support your report
  - Identify any problems or issues, but don't blame others

- Completion report
  - Used by the project team to make a go-live decision
  - Be factual
  - Include a list of open defects and tests that couldn't be run
  - It's a good idea not to express your opinion on whether or not the system should go live – just present the facts and let the project team decide.

Daily reporting

During test execution the test team should prepare and issue a report to the project stakeholders that includes:

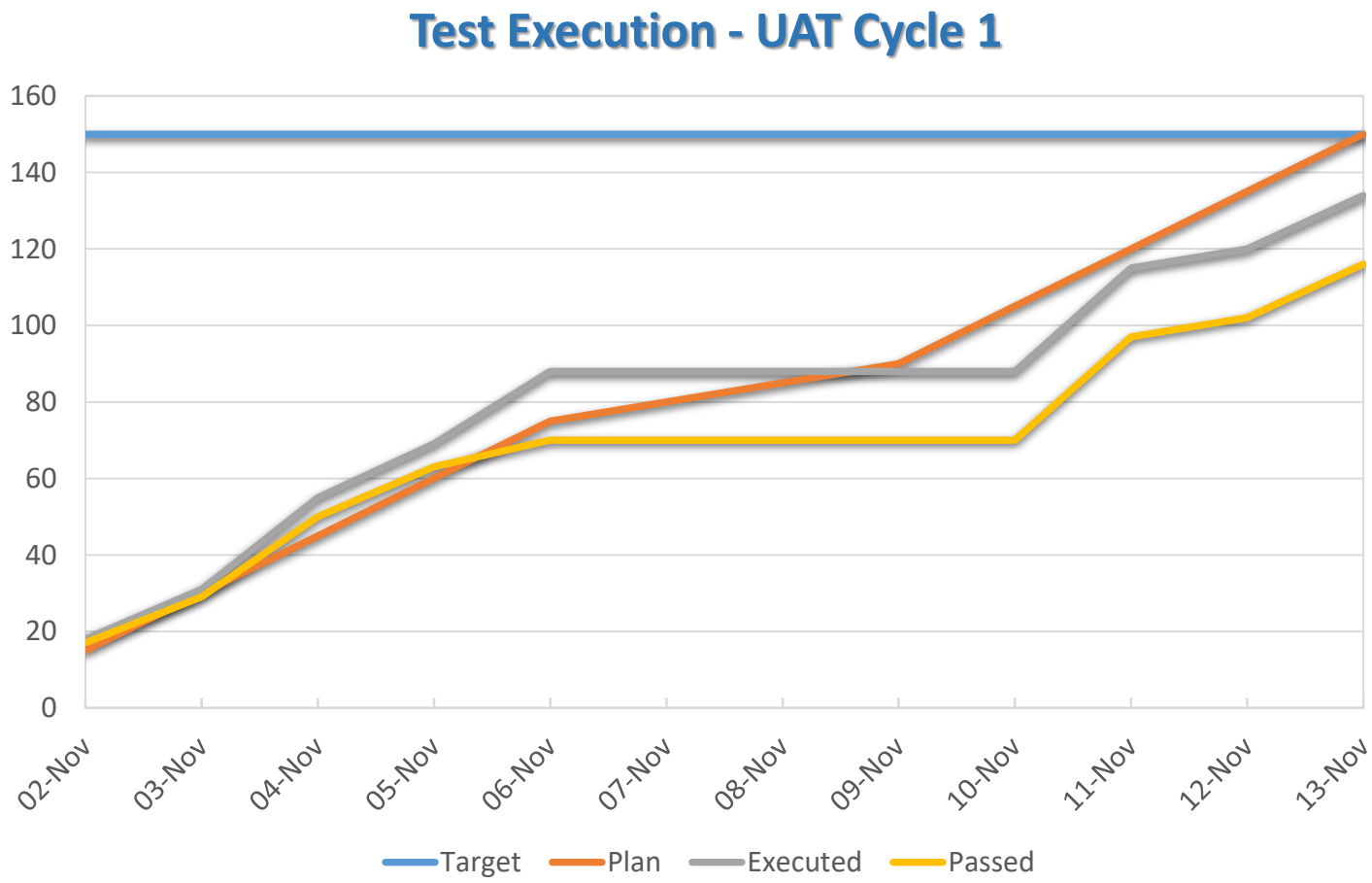
- A narrative summary of progress since the last report, any issues that have or are likely to impact testing progress and what activities are planned for the next period.
- Test environment status
- A summary of test progress to plan:
  - Total number of test cases planned
  - Total number of test cases executed, passed and failed to date
- A summary of defects found:
  - Total number of defects by status
  - Open defects by severity and priority
  - Number of defects raised and closed in the last day

But, try to keep it to one side of A4…

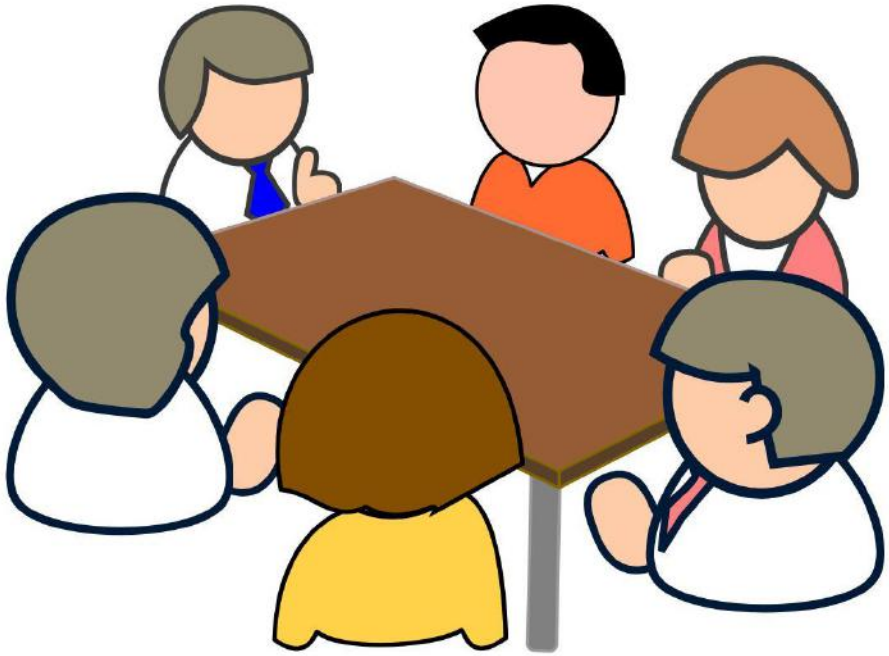
One-page reporting

- Test reporting should be kept simple and easy to understand
- Concentrate on information, not data
- Beware of creating a stick for others to beat you with:
  - Inaccurate numerical data – figures don't reconcile
  - Sudden changes in reported data – especially targets going up or down
  - Keep emotion out of the report and don't point the finger at others:
    - *The test environment was unavailable all day due to a configuration change being applied incorrectly* is unemotional and factually correct.
    - *The build team screwed up the test environment so we were unable to do anything all day* is probably not only factually incorrect, but invites others to put the blame on test another time.
  - Be honest if you or your team has got it wrong
- Report potential issues early, bad news doesn't improve with age!

Using charts in reports



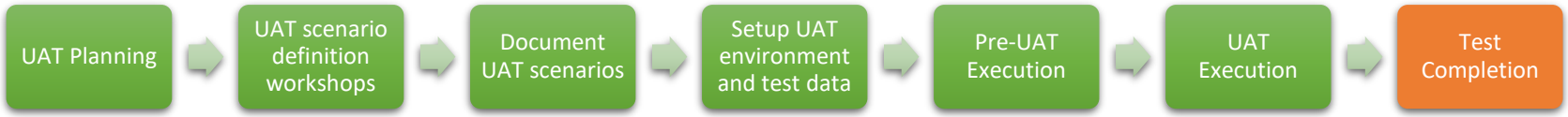
Exercise 9 – Reporting



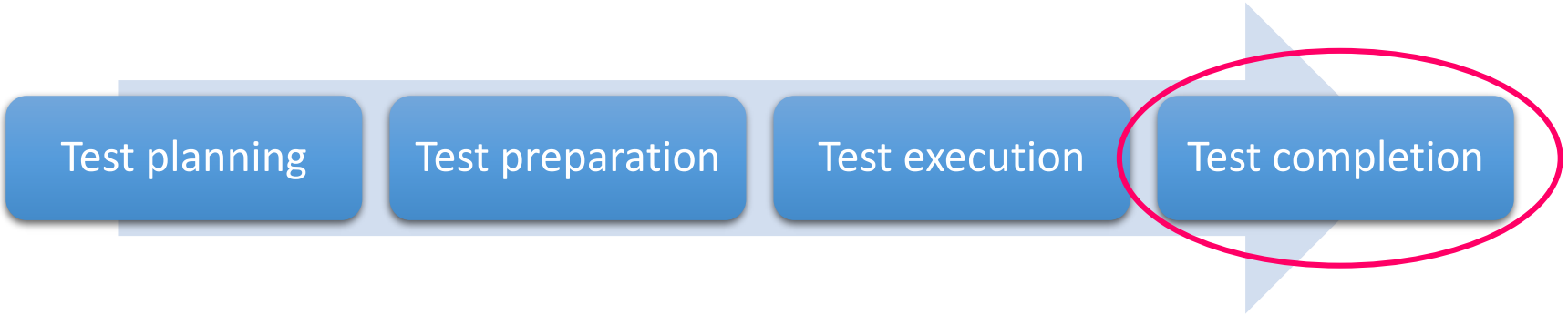
Yesterday, we were only able to complete 10 tests as the test environment was unstable during the morning following a code refresh and was unavailable for most of the afternoon while the environments team tried to resolve the issue.

How should we report this?

UAT completion



Completion report



At the end of UAT the test manager will prepare a completion report which will include:

- A narrative summary of what happened during the test phase.
- A summary of any issues found during testing
- A summary of the number of tests planned and completed:
  - Total number of test cases planned, executed, pass, fail and not run
- A list of all test cases that were not completed, with the reason why.
- A summary of defects found:
  - Total number of defects raised by severity and priority
  - Current number of defects raised by status
- A summary list of all open defects.

Test complete

Q&A

Introduction to User Acceptance Testing

TSG

Dawson House, 5 Jewry Street, London EC3N 2EX

www.testing-solutions.com

rgillan@testing-solutions.com (admin)

TestCon MOSCOW 2017

tsg

v1.3 TCM