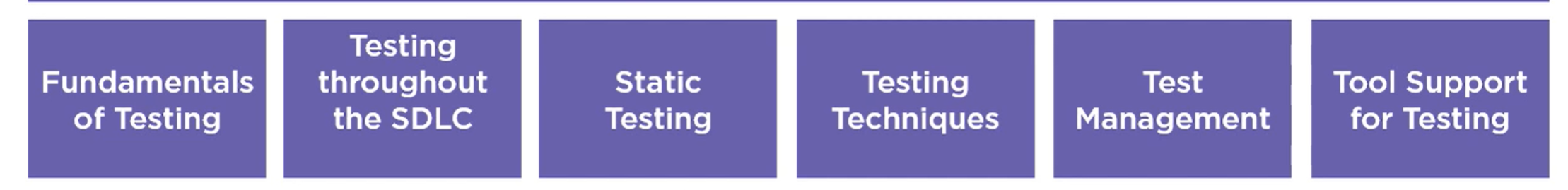
It doesn’t matter how good you are at automation if you don’t write the tests right

Structured quality tests to find bugs

6 topic:



Sub topic – what is testing and why is it necessary

One course for each chapter on pluralsight

What is Testing?

Why necessary?

Howe does it contribute to the end of product?

The seven testing principals!

The test process – organisation

The psychology of testing

Graphical user interface, text, application

Description automatically generated

Checking results

* Test planning
* - what how by when
* Do these requirements make sense
* How to design good or bad tests
* How to execute them
* Reporting bugs well
* Test reporting

Objectives testing

* Prevent defects by evaluating requirements
  + You want to get the bug BEFORE it happens, evaluate the requirements before the code is written and identify contradictions
  + Ask are all requirements logically sound?
  + Reacting not as good as preventative
* To find defects and failures and so reduce the risk of inadequate software
  + Somew flaws can be tolerated some can not
* Have all requirements been met?
* Check whether the tesxt object is complete and validated so it works as users/stakeholders expect
* To build confidence in ther quality of the object
* To provide sufficient information for stakeholders to make informed decisions about their software
  + Evey piece of software has bugs- you need to know what they are
  + Perfect is the enemy of the good- you need to be able to say to management what the bugs are so they can deliver an informed decision on whether to deliver and how to budget
* To comply with contractual, legal or regulatory standards
  + It is your job to ensure your software follows laws and standards and to prove that with testing

What testing is not:

* Testing is not debugging!
  + Tests shows failures caused by defects in the software – debugging is carried out by developers to find and fix the underlying cause
  + Basically testing is finding issues – debugging is finding the cause of the issue
  + In testin you say: Whsat you did, what happened, what should have happened

7 Major testing principals

* Test early, test often

**Need to be able to answer:**

* What is testing?
  + Includes many activities – a way of proving that things work, building trust in a product, communicating to managers the stage of a product development and ensure ing it meets standards given and that those standards are sensical
* What activities does testing involve?
  + Test planning: What, How, By When?
  + Analysis – Do these requirements make sense?
  + Test design – How to tell between a good or bad test
  + Executing tests
  + Bug reporting – what happened and under what circumstances
  + Test reporting – what does the test evidence that things work look like, what bugs are there and how serious?
* What are the typical testing objectives?
  + Preventing defects
  + Finding software defects
  + Verifying the requirements are met
  + Does the software work as expected?
  + Building confidence
  + Provide information – to allow for informed decisions
  + To ensure legal/contractual requirements are provably met
* What testing is NOT
  + Debugging

Module 2.

* Why testing is necessary?
  + Malfuction can cause costly/dangerous error
* Error vs. defect vs. failure
* Root cause of a fauilure vs it’s effects

Points for testing:

* Requirement review
* Works with system designers – make room for testing for each element
* A picture containing text, businesscard, screenshot

  Description automatically generatedA picture containing text, screenshot, businesscard

  Description automatically generated
* This way you can create backdoors for testing in the code
* Work closely with developers while they code
* Testers verify and validate code before release

**Know the difference for the exam!**

Quality assurance

Quality control

Testing

Quality management

* Quality assurance and quality control
* Quality assurance
  + Proper processed- are we following protocol?
  + When things go wrong – process to find root cause
  + Managing tool – proactice preventing defects
* Quality Control
  + Testing, test design, execution…
  + Find defects that you can’t prevent, reactive
  + Propertesting
  + QC is part of qa
  + Graphical user interface, application

    Description automatically generated

Error

* Mistake – human error, misunderstanding requirements for example
* Errors can lead to faults or defects

Defect

* Alswo known as a fault or bug
* Bugs can trigger further bugs
* Human mistake built into the software
* Bugs **may** lead to failure

Failure

* When you use the software and it doesn’t do what you want it to do
* **Ugs may only be manifest** after a specific sequence of events or accessed with a specific sequence of actions

Graphical user interface

Description automatically generated

REMEMBER dif for exam!

Why do mistakes happen?

* Time pressure
* Human fallibility
* In experience or lack of skill
* Complexity – more complex more mistakes
* Misunderstandings – never assume
* New or unfamiliar tech – someone has to trial and error

**Treat the cause, not the symptoms**

You need to find the cause of failures especially if it keeps happening

It could be that when the codce is reviewed/’fixed’ other things get broken or un checked

Qs. To ask yourself;

* Do you have enough skill for this job,
* Is there enough information between the stakeholders
* Root cause analysis MUST NEVER be a blame game, op to improve

Why important

* Leathl and costly, high maintenance cost
* Testers should be there for requirement review, working with system designers and software engineers
* Validating the software
* QA is NOT testing, QA is all tester activities with a focus on preventative measures
* QC is about product not process

Module 3

Seven Testing Principals

* 1. Testing shows the presence of defects not their absence
* 2. Exhaustive testing is impossible
* 3. Early testing saves time and money
* 4. Defects cluster together
* Beware of the pesticide paradox
* 6. Testing is context dependent
* 7. Absence of errors is a fallacy

1.

Follows the rule – **Absence of evidence is not evidence of absence**

NEVER claim a software has no bugs even if you are pushed to say so

You can say, I haven’t found any bugs, no further issues found

2.

Exhaustive testing

* Not practical or possible in most cases
* Near infinite options or combinations of options to try out
* Combinatorial explosion – is it possible to try out all combinations in the time allowed?
* You select a representative sample of all possible combinations instead, test, assume the rest is ok
* To select the portion use:
  + Risk analysis
  + Test techniques
    - Equivalence partitioning
    - Boundary analysis (BVA)
    - Decision Table
  + contextual priorities
  + Random selection is not advisible

3.

Earlier testing more efficient

4.

Defects cluter together

* Any non-trivbial system will be composed of modules
* Since the data you input travels through multiple modules it is bound to encounter defects
* Huh, Apparently in most cases a small number of modules contain most of the defects – this is the module you may need to focus on – helps you prioritise when you don’t have enough time
* Say you get a new feature, how do you prioritise testing? You test the new feature, then the modules with the most severe and frequent defects, then other things
* Prioritisation!

5.

Pesticide paradox

* Graphical user interface, text, application

  Description automatically generated
* This is the problem if you run the same tests over and over since they are unlikely to find anything new
* To find new defects you must:
  + Review and change existing tests and test data
  + Create new tests
* Text

  Description automatically generated with medium confidence
* Allows some variety in test data in scenario v2

6.

Context Dependent

What is important?

* May depend on time, location, stakeholder interests, etc.
* For example which app requires more security testing
* Test techniques and management – learning your system is important,but broader pic also important

7.

Absence of errors is a fallacy

- it may seem true, but it is not

- You may fix a lot of defects, it doesn’t mean your code is high quality

Durability testing

Testing shows the presence of defects not their absence (1), however absence fo visible errors does not mean the code is any code -you haven’t tested for everything

**Summary**

* You can prove the presence, but not absence of bugs
* Exhaustive testing is impossible so use analysis and techniques to test smart
* Early testings saves
* Defects tend to cluster in smaller subsystems, monitor where the errors appear and use it to test
* Vary your tests beware the pesticide paradox
* Your surroundings will affect your testing
* Absence of errors is a fallacy – you can increase the chances of good quality, but never guarantee success

Module 5.

There is no single way to test, however you can follow common high level test activities

External Factors which influence your testing:

* External Stakeholder Interests
* Domian in which you are working
* Laws and Standards

Internal team/organisation factors:

* Software Dev Lifecycles: Agile framework
* Budget/time/resources
* System complexity

Test Activities

* Be aware of thre context for priooritisation
* Test activities include:
  + |Test planning
  + Monitoring and control
  + Test analysis
  + Design
  + Implementation
  + Execution
  + Completion
* These steps donn’t need to be carried out in strict order

Test planning

* Define the objectives of testing and how you will achieve them
* Need to be specific
* What techniaues, how, what schedule
* Not set in stone

Monitoring and control

* On going comparison of planned against actual progress – are we on track to meet the deadline?
* Should we communicate to others if there are deviations from the plan?

Analysing the system

* The better you understand the system the better your tests will be, you analyse business, functional requirements, etc. ASnything with requirements, Design, code, risk analysis reports – all this provides a test basis
* When analysing these things you read them, identify defects in the requirements.
* Requirement defects means:
  + Ambiguities
  + Omiaaiona
  + Inconsistencies
  + Inaccuracies
  + Contradictions
  + Superfluous statements
* Analysios is a part of the early testing principal

Test Design

* You know what to test, now figure out how
* Design and prioritising cases
* Identifying necessary test data
* Prepare the test environment with necessary tools
* Ensuring bi-directional traceability between test basis conditions and cases

To Test:

1. You create test cases
2. You write the steps, expected results and input data, then group them into a test set,
3. write down what you need to test: tools, conditions, environment (browser version, software, etc.)
4. Map this to relevant part of the requirements so you can directly see how what your testing is relevant to the requirements
5. You may need to create a reqwuirements subset which specifies what is needed for testing conditions
6. You should create requirements to test mapping/traceability betw requirements, test basis, test products, etc.
7. Test design – how to test? Test implementation – Can we now runt the tests?

Implementation

* Developing test procedures and automated test scipts
* Setting up the test environment – VMs, etc.
* Preparing test data
* Verifying and updating tracability

Test Execution

* Set up as required by pre-conditions
* Compare expected and actual result
* If match YAY!
* If not, analyse and double check:
  + Was it set up correctly?
  + Did you follow the steps properly?
  + Does the test scenario have any faults?
  + Write a bug report and log it if you are sure your test is sound
* No matter the result you write a log of all outcomes, passed, failed, blocked, unable to run, etc.

Test Completion

* Done when a milestone is reached
* Includes:
  + Ensure all defect reports are closed
  + Terst summary report for stakeholders
  + Finalising and archiving or sharing testware – you may be required to do this
  + Analysing lessions learned

Test basis – reason for why you are testing and proving that you will be testing

Test Work Products

* Vary from organisation to org
* The product of your work, so for the test planning *activity* the work product is the test plan- test monitoring and control criteria – says when this then we are done
* Test monitoring and control activities give you test progress reports, test summaries
* Test analysis gives you Test conditions, defect reports
* Diagram

  Description automatically generated

Test condition – a testable aspect of a component or system identified as a basis for testing

**Summary:**

* No single correct test process, but some commonalities
* Context has a lot of influence
* Graphical user interface, text, application

  Description automatically generated

Module 6.

Test Psychology

Relevant human psych

Testers vs. developers

Testers MUST have interpersonal skills

* You are Reporting bad news
* You have to get someone else to admit to and fix a mistake, ego issues
* Also you may have spent a long time identifying and testing a bug – BUT you need to have the perspective to realise not all bugs will be worth fixing
* Aim is cooperation asnd communication
* Should not give overly critical feedback
* Should not overly report bugs and waste peoples time with false alarms

Ways to improve

* Develop a rapour with developers by talking to them
* Make it clear you are on the same side – delivering a working high quality product on time
* Here to support NOT critisise them
* You want developers to think oh thank god you found that bug!
* Fixing bugs early is great for all
* ALL communication should be clear, informative and constructive
* Clear bug report should answer:
  + What? When? How?
  + Software version tested
  + Link to the relevant requirements
  + Short description of what went wrong
  + Concrete steps to reproduce the bug
  + Expected result
  + Actual result
  + Screen shots, log snippets or other useful issue
* You should not cut corners here or the developer will find it harder or impossible to fix
* Read your own report and ensure you only provide relevant information
* Try to understand the other person, why might they react badly
* Make sure the other person has heard and understood – have them repeat back what you said
* More Resources:
* 

Tester vs. Developer

Graphical user interface, text, application

Description automatically generatedDiagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

This is why you need testers and developers to be different

Desired traits in testers includew:

* Curiosity
* Professional pessimism
* Critical eye
* Attention to detail
* Mostivation for good positive communications and relationships

Graphical user interface, text, application

Description automatically generated

**Summary for Course 1:**

* What? Why?
* Testing is not debugging
* Done throughout the dev lifecylcle
* As a tester you drive quality management – qa and qc included
* Error -> Debug ->Failure
* 7 Principals
* Test process fundamentals from planning and analysis to execution and completion with associated work products
* Importance of soft skills