Notes from 15 Aug

- **Learn testing principles and process**
- **objectives of testing throughout dev lifecycle**
- **apply a variety of test techniques**
- **understand the need for testing**

Risk and quality:

- Justin's three things
  - legal impact to the company
  - Financial impact
  - Brand impact(reputation)
- Seven testing principles
  - Testing shows the presence of bugs
  - Exhaustive testing is impossible
  - Early testing is important
  - Defect clustering(Several issues in the same areas)
  - pesticide paradox
  - testing is context dependant
  - Absence of errors fallacy
- Key testing terms
  - defect
  - effects a defect might have on a project
  - root cause vs efect
  - error
  - failure
  - fault
  - mistake
  - bug
- Debugging and testing
  - Debugging lives more with developers and is used to identify the cause of bugs or defects in code and undertake corrections
  - Testing systematic exploration of a component or system with the main aim of finding and reporting defects

Fundamental test process

- planning and control
  - Determines what is to be tested
  - how it is going to be tested
  - who will do it
  - define our exit criteria
  - monitoring and control feed back into planning
- analysis and design
  - Review the test basis
  - identify conditions cases and procedures
  - design tests
  - detail environment and tooling
  - highlight the test data
  - create traceability between basis and cases

- implementation and execution
  - prioritising test cases,creating data and writing procedures
  - creating test suites
  - checking environment is set up correctly
  - running tests in determined order
  - log testing activities and defects -depends on the model
  - reporting incidents
- exit criteria and reporting
  - ensure exit criteria been met
  - determine if more tests needs
  - checking the system is in a read state for release
  - writing up results for sponsors and stakeholders
- closure activities
  - make sure docs are up to data and archived
  - closing down and archiving the test environment infrastructure and test ware
  - passing over test ware to maintenance teams
  - documenting the lessons learnt

Development models

- Waterfall model
  - system requirements \/
    - software requirements \/
      - analysis \/
        - program design \/
          - coding \/
            - testing \/
              - operations
- V model
  -
  - requirement spec acceptance testing
    - functional spec system testing
      - tech spec integration testing
        - program spec unit testing
          - coding
  - write a unit test before coding
  - lots and lots of documentation
  - verification and validation
    - verification checks the work meets the requirements
    - helps to insure building the right way
    - validation evaluates against user needs rather than work product
    - ensure the behaviour of the product meets the needs
    - helps to make sure we are building the right product for our users
- iterative and incremental models
  - >>**requirements** >>>> **design**\/
  - /**test** <<<<<<<<<<**code**<<
  - SCRUM, lots of sprints of separate areas of code
  - sprint backlog sits in requirements

Test Levels

Apply different test levels at appropriate stages
recognise and compare functional, non-functional , structural and change related testing
describe the purpose of confirmation and regression testing
analyse software from a structural and architectural perspective
identify reasons for maintenance testing and the role of testers

- **Unit testing**
  - **Units are also called programs modules or components**
  - **code is written in component parts or units constructed in isolation amor integration at a later state**
  - **each of those units of code will relate to a certain function and will be tested by the developer**
  - **unit testing is likely to be on small functions that do a highly defined task**
  - **test driven development- tests first**
  - **fail refactor fail refactor fail refactor pass**
- **integration testing**
  - **Purpose is to expose defects in the interfaces and interactions between integrated components or systems**
  - **the test objects are essentially the interface code**
  - **three types of integration testing**
    - **big bang**
      - **all units are linked at once resulting in a complete system**
      - **difficult to isolate a specific area of error**
      - **introduces risk that problems may be found at a later date**
    - **top down**
      - **high level modules are created first so a stub code is needed**
      - **stubs give the illusion of functionality to fill in where code doesn't exist yet**
      - **useful creating generic software**
      - **allows for early demos of functionality**
      - **may help identify requirement changes and issues**
      - **stubs can create a lot of work**
      - **stub definition can be difficult**
      - **reproducing test conditions may not be possible**
    - **bottom up**
      - **in the absence of parent modules you use drivers**
      - **higher accuracy at smaller levels**
      - **components are added in a controller manner**
      - **user and business awareness used to clarify product in early stages**
      - **difficult to estimate top level forecasts**
      - **driver system can create a bigger workload**
- **system testing**
  - **test functionality from end to end**
  - **focuses on a whole system in a live environment**
  - **carried about by an independent team**

    - tests functional and none functional requirements
- acceptance testing
    - contract and regulation acceptance testing
    - alpha and beta acceptance testing
    - user acceptance testing
        - tested by user rep
        - checks the system meets business needs
    - operational acceptance testing
        - checks processes and procedures are in place to allow the system to be used and maintained. includes checking: back up facilities, procedures for disaster recovery, training for end users, maintenance procedures, data load and migration tasks and security procedures
- types of testing
    - functional
        - testing the functions of a system verifying a specific action or function
        - sometimes called specification testing, testing against the spec
    - non-functional
        - testing characteristics
        - install-ability
        - maintainability
        - performance
        - load handling
        - stress handling
        - portability
        - recoverability
        - reliability
        - usability
    - structural(White box)
        - how the code makes the functionality work (Under the hood)
            - any test level
            - decision coverage
            - statement coverage
    - change related maintenance test
        - testing in a live environment
            - modification
            - migration
            - retirement of software
        - impact analysis(risk) and metrics
        - they help estimate amount of re-testing and regression testing
        - what are the possible consequences
        - what areas will remain unchanged
    - regression testing
        - carried out on every other part to ensure a fixed defect hasn't created bugs elsewhere
        - repeated testing of already tested program
        - performed when software or environment is changed
        - based on risk

        - **performed at all levels(functional, non functional, structural)**
    - **RCRCRC**
        - **recent**
        - **core**
        - **risk**
        - **configuration**
        - **repaired**
        - **chronic**

test design

- **understand why test design techniques exist and what a test basis document is**
- **differentiate between test design spec , test case spec and test procedure spec**
- **Agile - loose documentation wise, everything else has lots**
- **regulatory or safety testing requires more documentation**
- **consider context**
    - **organisation**
    - **maturity of dev and test process**
    - **time constraints**
    - **safety or regulatory requirements**
    - **the people involved**
- **test design spec**
    - **identifier**
    - **features**
    - **approach**
    - **test identification**
    - **pass fail criteria**
- test terms
    - test base- documents where the requirements are
    - test conditions- characteristics which can be tested
    - test cases - a set of input values
    - test procedures- actions for the execution of a test
- Test cases
    - set of input values
        - execution preconditions
        - expected results
        - execution post-conditions
    - developed to cover a certain test objective or test condition
    - test conditions are what are we going to test, test cases are how are we going to test
    - did this in college id , precondition, data, expected, actual , pass/fail , post condition

Conditions

- can register
- suitable protection on registration fields
- validates on email
- does receive emails

cases

- tc1, registration form, valid email +pass , successful reg, ****** , *****, ******
- tc3 registration form , invalid email + pass, unsuccessful reg, **** ,****,***
- tc4 registration form, invalid email valid pass, unsuccessful reg, *****, ****,****
- tc5 reg form, valid email invalid pass, unsuccessful reg, ****,****,*****
- tc2, set up emails, set to a certain period, periodic emails sent, ****, **** ,****
- tc7, registered to database

websites beginning to struggle with email validations

Test procedures (Scripts)

- Identify all actions in sequence to execute a test
- often called test scripts
- execute a test case by inputting the values and checking the outcome
- e.g launch browser, move mouse to log in panel, enter username: tstation1, move mouse to password panel

TC1

1. **go to spartaglobal.com**
2. **go to register link**
3. **move mouse to email field**
4. **enter email : tst1@sbg.com**
5. **move mouse to password field**
6. **enter email : pa55word**
7. **click submit**

TC3

1. **go to spartaglobal.com**
2. **go to register link**
3. **move mouse to email field**
4. **enter email : tst1**
5. **move mouse to password field**
6. **enter email : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**
7. **click submit**

TC4

1. **go to spartaglobal.com**
2. **go to register link**
3. **move mouse to email field**
4. **enter email : tst1**
5. **move mouse to password field**
6. **enter email : pa55word**
7. **click submit**

TC5

1. **go to spartaglobal.com**

2. **go to register link**
3. **move mouse to email field**
4. **enter email : tst1@sbg.com**
5. **move mouse to password field**
6. **enter email : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**
7. **click submit**

TC2

1. **go to spartaglobal.com**
2. **go to log in link**
3. **move mouse to email field**
4. **enter email : tst1@sbg.com**
5. **move mouse to password field**
6. **enter email : pa55word**
7. **go to settings**
8. **move mouse to email notification section**
9. **click for periodical emails : twice weekly**
10. **click submit**

TC7

1. **log in to database server**
2. **move to database search**
3. **enter tst1@sbg.com**
4. **view results**

Harvey

- Every engineer is involved in testing

Test coverage

- quantitive measures are important to see home much has been tested
- coverage might make up part of competition criteria it can also be used to tell us when to stop testing

standardisation: write in keeping with existing documentation
build a tool kit of documentation templates, adjust and improve the spec, test cases and test procedures in practical sessions

SBT - specification based techniques

- examines functionality of an application without knowledge of internal structures
- it is based entirely on the software requirements and spec
- spec can include non-functional
- examples
    - sanity/smoke check
        - smoke test
            - not to find defects but to check system health
        - makes sure bugs reported in previous builds are fixed before doing a

full regression test
        -
    - equivalence partitioning
    - boundary value analysis i.e. should take value up to 200
    - decision table testing
        - list all input conditions
        - each column represents a single business rule
        - given in bool values
    - state transition testing, logged out logged in
    - use case testing UML
- refer to decision chart on excel
- infeasibility
    - certain combinations of conditions that are logically impossible
    - we may want to test infeasibly test condition to ensure it can't happen
- state transition testing
    - testing of different states that the application can be in
    - logged out/in transition
    - transformation are determined by the rules of the system
    - this means we can follow these rules to create a diagram that represents the change of transitions and thus a test to see if it works
    - GOAL: finding situations where the wrong action or wrong new state occurs in response to a particular event
    - as testers we report these problems all the time as defects
    - terminology
        - states (Door open/closed)
        - transitions (opening/closing)
        - inputs or events (Twist handle)
        - actions(door is open/closed)