Software Testing Interviewing Questions

- PDCA (Plan, Do, Check, Act)
- Box Testing
    - Black Box – solely on requirements and specifications
    - White Box – internal paths, code structures, and implementation of software
    - Grey Box – Look at code to understand implementation, then test via black box
- Defect v.s. Failure
    - Defect found in-house
    - Failure – client found
- Defect Categories
    - Wrong – variance from requirement
    - Missing – requirement that isn't coded
    - Extra – requirement missing from specification
- Verification v.s. Validation
    - Verification – review without execurint the process
    - Validation – checking product with actual execution
- Does increase in testing always improve project?
    - No, 20% of test plan is critical
- Defining Testing Policy
    - Definition – one unique definistion for testing
    - How to achieve – testing committee/test plans?
    - Evaluate – how to derive metrics of defects (phase/programmer). How has testing added value
    - Standards – what is the standard of good enough
- Should testing be done after build/execution phases?
    - Traditional (requirement → Design → Code & Build → Testing → Maintenance)
    - Modern
        - Requirement → Design → Build & Execution
        - Test → Installation → Maintenance
- More Defects in Design or Coding
    - Design!! (requirements, then bad architecture & technical decisions)
    - 60% defects occur in design, 40% in coding
- End User Input for proper testing
    - Acceptance test plan prior to production
    - Requirement documents signed by client
    - Risky sections of project, obstacles to use
    - Proper data for testing
    - Test scenarios
- Latent and masked defects

    - Latent – existing defect that has not yet caused a failure due to conditions not met
    - Masked – existing defect that hasn't yet caused a failure due to another defect prevented code execution
- Defects not removed in initial stages can be up to 20 times more costly to fix in maintenance phase
- Workbench concept
    - Documenting how an activity must be performed (phases, steps, tasks)
    - Workbench
        - Input – input/entrance criteria
        - Execute – transform input to expected output
        - Check – assure output meets desired result
        - Production output – exit criteria of workbench
        - Rework – once fix implemented, go to execute
    - Phases

	Input	Execute	Check	Output
Requirement	Customer requirements	Write requirement doc	Addresses all needs?	Requirement Doc
Design	Requirement Doc	Write Technical Doc	Technically correct?	Technical Doc
Execution	Technical Doc	Implementation & coding		Implementation & Source code
Testing	Source code	Test cases		Test results
Deployment	Source code/test results			Production
Maintenance	Deployment results, Change Req		Regression Testing	New release

- Alpha(development) and Beta (Customer location)
- Defect leads to other defects based on their dependency/inheritance of object
- Usability testing
    - Give client prototype or mock-up, generally only the UI portion of project
- Strategies for Rollout to End Users
    - Pilot – limited deployment to a control set and is usually considered Beta Test
    - Gradual Implementation – deploy entire product while working on a finished product. Must maintain multiple versions
    - Phased Implementation – Rollout to all users incrementally as features are completed
    - Parallel Implementation – existing and new applications run simultaneously, but need duplicated hardware

- Requirement Traceability
    - Testing begins at requirement phase
    - Ensures proper coverage
    - Easily identify project debt as defects found
    - Identify risk with current state of software
- Pilot v.s. Beta Testing
    - Pilot product installed at live locations(Entering Real Data)
    - Beta testing not done with real data (Acceptance Test Plan)
- Rick Analysis During Testing

Features	Probability of Failure	Impact	Priority
Add a user	Low	Low	1
Check user prefs	Low	Low	1
Login User	Low	High	2
Add new invoice	High	High	4
Print invoice	Med	High	3
Concerns			
Maintainability	Low	Low	1
Security	High	High	4
Performance	High	Low	3
	Low/Med/High	**High/Low**	**1-4**

    - List Features/Concerns
    - Rate Defect probabilities
    - Impact Rating
    - Compute Risk(Probabilty * Impact)
    - Review
- Acceptance Plan Preparation
    - Requirement document
    - Input form the customer
    - Project Plan (project manager)
    - User Manual
- Environment reality and Test Phases
    - Latter Test Phases need more reality
- Inspections and Walkthroughs
    - Walkthrough is informal
    - Inspection is formal and documented
- Confirmation v.s. regression
    - Confirmation verifies fix works
    - Regression verifies nothing else broke as a result
- Coverage v.s. Techniques
    - Statement coverage – source code completed covered

    - Decision c overage – ensures Booleans covered
    - Path coverage – verifies that every possible route through code is covered
- Coverage Tools
    - Run simultaneously with testing and reports what was tested and results
- Configuration Management
    - Detailed recording and updating of infor for hardware and software components(requirements, design, test cases)
    - Knowing the state of the test system and expected outputs
- Baseline concept
- Different test Plan documents
    - Project Test Plan – resource utilization, testing strategies, estimation, risk, priorities
    - Acceptance Test Plan – verify user requirements are met
    - System Test Plan – main testing, functionality, load, performance, reliability
    - Integration Testing – tying software and hardware components together and verifying interoperability
    - Unit testing – developer level to check individual module in test

	Project TP	
Integration TP	Central TP	Acceptance TP
Unit TP		System TP

	Central				
Requirement		Acceptance			
Design			System		
Execution				Integration	Unit
Testing					
Deployment					
Maintenance					

- Inventories, Analysis & Design for Testing Projects, Calibration
    - Test objectives
        - Policies
        - Error Checking
        - Features
        - Speed
    - Inventory
        - List of things to be test for an objective
        - Add new Policy
        - Change/Add Address
        - Delete a Customer
    - Tracking Matrix
        - Inventory ← Mapping → Test Case

        - Mapping is Calibrating inventory to test cases
- Black Box or White Box Testing first?
    - Black Box first
    - White Box only after Design is fairly stable
    - Requirement Doc, Design Doc, Project Plan → Black Box → White Box
- Cohabiting Software
    - Software that can reside on same PC and affect project (and vice versa)
- Impact Rating used (Minor/Major/Critical)
- Test Log = data produced from test case(s)
- SDLC
    - Entry Criteria → Estimation Doc
        - SDLC
            - Exit Criteria
                - Acceptance Doc
    - WaterFall
        - Big Bang Waterfall
            - Requirement
            - Design
            - Build
            - Test
            - Deliver
        - Phased Waterfall
            - Project divided and developed in parallel, gluing things together at the end. Problems arise with lack of coordination
    - Iterative
        - Incremental – work divided into chunks like the Phase Waterfall, but one team can work on one OR many chunks
        - Spiral – Prototype and refining cycle, waterfall is repeated for each cycle
    - Evolutionary
        - Produce minimal features using process, then evolve software with updated customer requirements
    - V-model
        - Emphasizes importance of early testing so that each Stage is refined immediately after introduction
- Testing phases
    - Unit – Component Design Features fully covered, usually done by a developer by creating fake components that mimic what finished components due in order to test the logic
    - Integration – tests all components and their interoperability/relationships
    - System – System Specification driven and tests the entire system as a whole.
        - Performance

        - Volume
        - Stress
        - Documentation
        - Robustness
    - Acceptance testing
        - Checks system against requirements, done by customer NOT developer
- Who does testing?
    - Isolated test Team
    - Outsource
    - Inside
    - Developers as testers
    - QA/QC Team
- Equivalence Partitionign
    - Eliminating duplicate test cases for maximum efficiency
    - Test <, >, and =
    - All redundancies should be eliminated
- States and Transitions
    - State = result of previous input
    - Transition = actions that cause change in state
    - Each state or transition can generate test cases
    - Combine state and transition for complete coverage since using only one in isolation can leave gaps in coverage
- Random/Monkey testing
    - Not realistic
    - Redundant
    - Time spent analyzing results
    - Cannot recreate the test if data isn't recorded
- Negative/Positive testings
    - Negative – invalid input = errors
    - Positive – valid input with some expectation on output
- Exploratory Testing (Adhoc)
    - Unplanned, unstructured, impulsive or intuitive journey with intent to find bugs
    - Simultaneous learning, test design, and test execution
    - Any testing doen tot eh extent that the tester proactively controls the design of the tests as those tests are performed and uses info gained while testing to design better tests.
    - NOT random
    - Learning → Design → Execution
- Semi-Random
    - Random testing that removes duplicates

    - Random test case → Equivalent Partitionign → Semi-Random Test Case
- Orthogonal Arrays / Pair-wise defect
    -
-
    -