Testing Software Quality Characteristics

Performance Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Performance Testing - ISO 29119

Performance testing

- Is aimed at assessing the performance of the test item
- When it is placed under a “typical” load

Performance Testing

- Verify system meets its performance requirements
- Quantitative and measurable performance requirements
- Reasonable stable system
- Test environment representative of customer site
- Tools for load generator

Performance Testing

Define performance requirements

- Quantitative
- Measureable
- For specific functionality
- Related to use case
- Under what circumstances/condition

Load Specification

How do we specify load?

- Identify relevant use-cases
- Load representing volumes of an activity
- Load representing mixes of activities

Testing Software Quality Characteristics

Stress Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Stress Testing - ISO 29119

Stress testing

- Is aimed at assessing the performance of the test item
- When pushed beyond its anticipated peak load
- Or when available resources (e.g. memory, processor, disk) are reduced
- Below specific min requirements
- To evaluate how it behaves under extreme conditions

Stress Testing

How do we conduct stress test

a. Identify stress points - **CPU, memory, buffer**
b. Develop a strategy to stress the system at stress points- **use tools to generate load, simulations**
c. Verify the generation of stress - **is it really stressing, modify strategy**
d. Observe behaviour - **stress requirements are met, functionality is correct**

Testing Software Quality Characteristics

Volume Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Volume Testing

What is volume test

- Verify the behaviour of system meets its requirements (functional and performance)
- When the system is subjected to a large volume of activity
- **Over an extended period of time - 24/7**

Volume Testing

What can happen when systems are used extended periods of time?

- Memory Leak - memory filled up
- Counter overflow -
- Resource depletion - resource used faster than it can be recovered

Testing Software Quality Characteristics

Configuration Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Configuration Testing Steps

- **Identify the parameter that define each configuration**
  - that could have an impact on the system's ability to meet its functional and performance requirements
- **Partition (group similar configurations)**
- **Identify configuration combination to test**
  - Extremes (min and max)
  - Risk based
  - Design of experiments (pair wise)

Testing Software Quality Characteristics

Regression Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Regression Testing

What is regression testing?

- Ensure that previously introduced and tested functions continue to work as specified after software modifications

What is the strategy?

- Run regression tests at multiple levels:
  - Unit level
  - Integration level
  - System level

Regression Testing

Who run regression testing?

Depends on the strategy

- Run regression tests at multiple levels:
  - Unit level - developers
  - Integration level - testers
  - System level - testers

Selective Regression Testing Strategy

Selective

- Rerun selected subset based on modification
  - Requires communication with developer - what is changed
  - Requires tools finding code deltas, ripple effect analysis
  - Use AI
- Rerun selected subset regardless of the modification,
  - Standard Confidence Test (Basic Acceptance Test - BAT)

Selective Regression Testing Strategy

Tools for Selective Regression Testing

- Code Deltas - keep track of Deltas (what is changed)
  - SCM - Software Configuration Management
  - Coverage Tools
  - Rerun test that traversed changed or deleted code
- Ripple Effect
  - Impact of modification to other features/code - **who does this - developers**
  - Communication of impacted features/requirements - **to who - test team**

Selective Regression Testing Strategy

What is checked by the confidence test?

- Confidence test is a subset of tests that is **executed to verify previous functionality**

Selective Regression Testing Strategy

How to select tests for confidence test?

- High frequency use cases
- Critical functionality
- Functional breath

Selective Regression Testing Strategy

- **Can we use the same regression tests in every release?**
- **Can we use the same confidence tests in every release?**
  - Both needs update
  - They need to be revalidated (care and maintenance)

Testing Software Quality Characteristics

Reading - Model Based Regression Test Selection Technique

Ira A. Fulton Schools of Engineering
Arizona State University

Operation and Maintenance Phase

Modifications in code - Delta

No modifications in code - in operation and maintenance phase

- Used in a different way versus initial context
  - Security and privacy treats
  - Safety impact
  - Hazards introduced by system changes
  - Performance needs changed
  - Environment hardware changes

Testing Software Quality Characteristics

Mobile Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Mobile Testing

How mobile testing different from web application?

- Input
- Context aware - location, time, user depended
- Connectivity - network (many different networks)
- Security and reliability
- Usability - GUI testing
- Diversity of operation systems hardware - configuration, use emulators and simulators, crowd testing, exploratory testing
- Additional hardware - touch screen, voice recognition
- Languages
- Resource usage - stress and load testing

Testing Software Quality Characteristics - 2

Error Detection, Recovery and Serviceability Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Reliability

- defined as the probability that a system will produce correct outputs up to some given time. [1]
- Reliability is enhanced by features that help to avoid, detect and repair hardware/software faults.
- A reliable system does not silently continue and deliver results that include uncorrected corrupted data. Instead, it detects and, if possible, corrects the corruption, for example: by retrying an operation for transient (soft) or intermittent errors, or else, for uncorrectable errors, isolating the fault and reporting it to higher-level recovery mechanisms (which may failover to redundant replacement hardware, etc.), or else by halting the affected program or the entire system and reporting the corruption.[2]

[1]E.J. McClusky & S. Mitra (2004). "Fault Tolerance" in Computer Science Handbook 2ed. ed. A.B. Tucker. CRC Press.
[2] https://en.wikipedia.org/wiki/Reliability,_availability_and_serviceability

Availability

- the probability that a system is operational at a given time, i.e. the amount of time a device is actually operating as the percentage of total time it should be operating.
- High-availability systems may report availability in terms of minutes or hours of downtime per year.
- Availability features allow the system to stay operational even when faults do occur.
- A highly available system would disable the malfunctioning portion and continue operating at a reduced capacity.
- In contrast, a less capable system might crash and become totally nonoperational.
- Availability is typically given as a percentage of the time a system is expected to be available, e.g., 99.999 percent ("five nines").

https://en.wikipedia.org/wiki/Reliability,_availability_and_serviceability

Serviceability/Maintainability

|the simplicity and speed with which a system can be repaired or maintained;
|if the time to repair a failed system increases, then availability will decrease.
|Serviceability includes various methods of easily diagnosing the system when problems arise.
|Early detection of faults can decrease or avoid system downtime.

- For example, some enterprise systems can automatically call a service center (without human intervention) when the system experiences a system fault.
- The traditional focus has been on making the correct repairs with as little disruption to normal operations as possible.

|https://en.wikipedia.org/wiki/Reliability,_availability_and_serviceability

Error Detection and Recovery Testing

- **System reliability and availability is dependent upon system's ability to**
  - Detect failures
  - Recovery from failures
- **What kind of failures?**
  - User
  - Hardware
  - Software
  - Other systems

Error Detection and Recovery Testing

System reliability and availability is dependent upon system's ability to

- Detect failures
- Recovery from failures

How?

- It is essential to have lists of errors to recover from which are listed in the requirements
- Inject error
- Detect failures
- Recover from failures

Serviceability Testing

System Serviceability Requirements

- What might a serviceability requirement include?
  - How problem is reported
  - How it is isolated
  - How correction should be made
  - How correction should be verified
  - Which release contains it

Serviceability Testing

System Serviceability Requirements

How do you test serviceability requirements?

- Inject failures - what kind of failures are defined in the requirements
- Observed the results - expected behaviour should be defined

Students, write your response!

Testing Software Quality Characteristics - 2

Usability Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Usability Testing

Usability of a system

Who are involved in this?

- Human systems engineers
- Human computer interaction designers/engineers
- UX Designer
- Information Architect
- Usability Tester

Usability Testing

What is usability of a system?

- Usability is the degree which intended users are:
  - able to perform tasks the product is intended to support
  - In intended environments
  - Satisfaction of the user
  - Users are protected

Usability Requirements

Usability requirements are stated in terms of what?

- **Learnability** - how much time and effort it takes to bring user to a desired level of performance
- **Memorability** - how intuitive, retain skills in using a product once it is learned
- **Errors** - number of incorrect interactions a user makes in trying to accomplish a task
- **Efficiency** - speed in completing tasks
- **Subjective satisfaction** - overall satisfaction

Usability Labs

When creating usability labs, what are some of concerns we have?

- How reliable the test results
- What is the validity of the test results
  - Wrong user - work with best users?
  - Wrong tasks

Usability Labs

What is the goal of usability testing?

- Formative Evaluation
  - Which aspects of interface are good and bad
  - How design can be improved
- Summative Evaluation
  - Assess overall quality of the interface

Usability Labs

When usability testing be done?

- Not end of product development
- Early in the design process with various mockups
- Pilot tests

Usability Testing

What should we consider when planning usability testing?

- Who are the users - representatives of the user (novice/experts)
- What tasks they will perform
- What kind of training
- What user aids will be available
- What data will be collected
  - Usability of product by different users
  - Usability of product by same type of users
- What criteria will be used to determine success
- How do we collect data - ethical aspects

Usability Testing Stages

- Preparation of environment
- Introduction
- Run test
- Debriefing

Usability Testing Techniques

Think Aloud

- What happens when user is silent, what do you do?
  - Periodically prompt test subject and remind them

Testing Software Quality Characteristics - 2

Reliability Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Reliability

Definition of reliability

- The probability of a system or a capability functions
- without failure
- for a specified time or
- number of natural units (e.g. number of calls, etc)
- in a specified environment

Availability

Definition of availability

- The probability at a given time that a system or
- Capability of a system functions
- Satisfactorily
- In a specified environment

Calculation of SW Availability

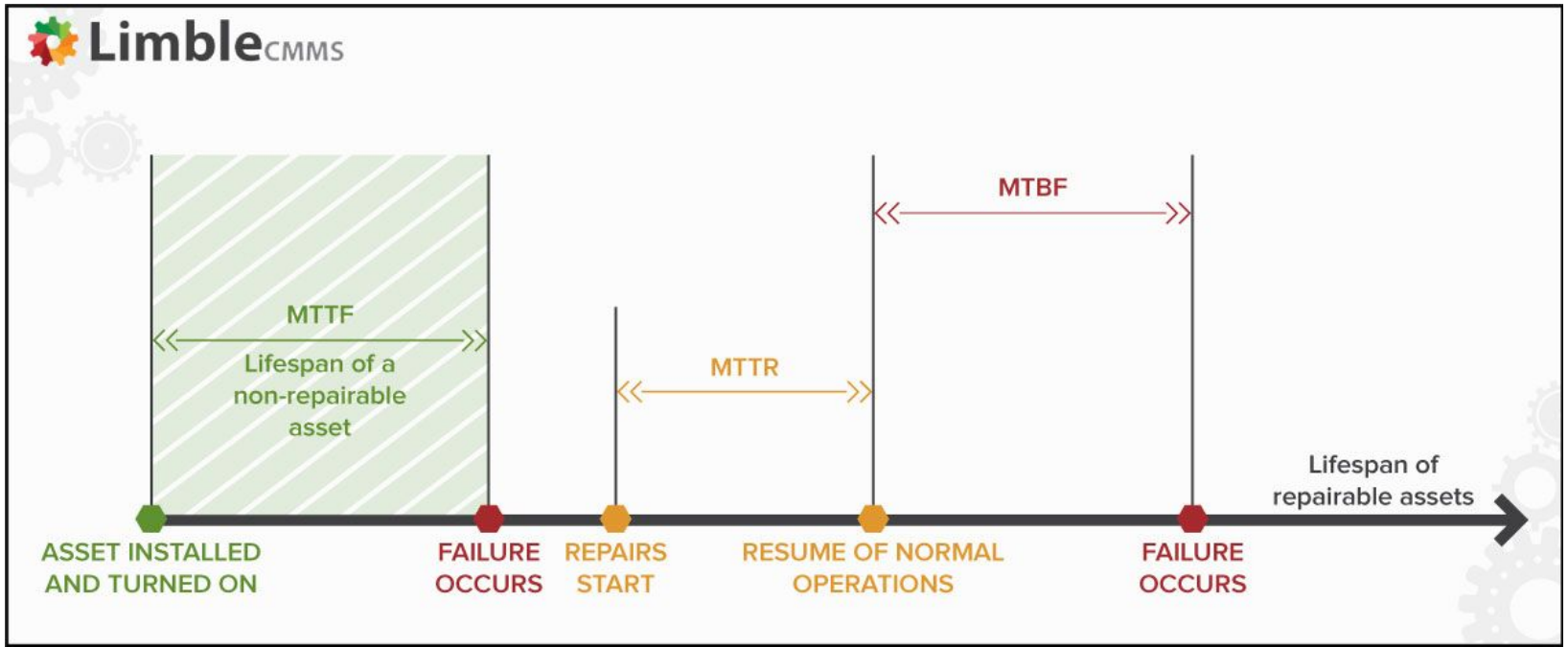
- Availability - percentage of time the system performs satisfactorily

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

- MTTF - Mean Time to Failure
- MTTR - Mean Time to Repair

Calculation of SW Availability

- MTTR (Mean Time To Repair)
- MTBF (Mean Time Between Failures)
- MTTF (Mean Time To Failure)



Calculation of SW Availability

- Availability?
- High MTTF - Mean Time to Failure - failure occur late
- Low MTTR - Mean Time to Repair - make fixes fast

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

Calculation of SW Availability

- Availability?
- Low MTTF - Mean Time to Failure - failing frequently
- Low MTTR - Mean Time to Repair - make fixes fast

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

Calculation of SW Availability

- Availability?
- Low MTTF - Mean Time to Failure - failing frequently
- High MTTR - Mean Time to Repair - make fixes slow

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

5Nines

What is 5NINES availability requirement?

- The system is available 99.999 % of the time

The service to a customer over a year is how many minutes?

- 5 minutes
- One minute of downtime per 100,000 minutes (there are 525,600 minutes per year)

5Nines

How do you achieve high reliability and availability?

- Use of SW Reliability Engineering techniques
  - Fault prevention
    - N-version programming
    - Redundancy
  - Fault tolerance

Students, write your response!

Achieving high reliability and availability

What Testing techniques are used?

Appropriate testing techniques must be applied along with models of assessing whether reliability and availability objectives are met

- Operational profile testing
- Error detection and recovery testing
- Serviceability testing

Operational Profile

What is an operational profile?

- Describing how users utilize a product or service
- Test the product based upon how it is going to be used by different users
- Operation profile consists of
  - Set of users
  - set of major functions
  - performed by system and
  - their occurrence probabilities

Students, write your response!

Operational Profile Steps

How do we use operational profile in testing?

- Develop test cases based on operational profile
- Modify test cases to incorporate critical functions with low occurrence probabilities
- Decide on the number of tests to run based on reliability objectives

Students, write your response!

Operational Profile Steps

How do we interpret failure data

- Development testing - remove faults that are causing failures
- Certification testing - determine whether a software or system can be accepted or rejected.

Operational Profile Steps

How do we use operational profiles?

- Assist performance analysis
- Guide development priorities
- Reliability models take operational profile

Students, write your response!

Testing Software Quality Characteristics - 2

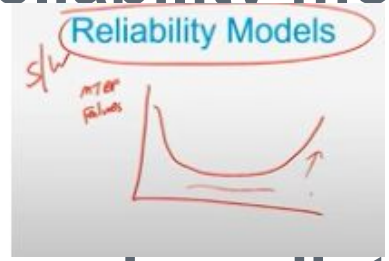
Reliability Models

Ira A. Fulton Schools of Engineering
Arizona State University

Reliability Models

Hardware versus Software reliability model

- Bathtub model in hw
- No burn down of sw
- Reliability improves in sw



Failure data fed into models and predict

- What was observed
  - Different models had different prediction

Reliability Growth Models

Reliability changes over time

- Assume that reliability improves over time
- Why?
  - Due to test debug and remove defect cycle

Reliability Growth Models

- **Models support when to stop testing**
- **Effectiveness of collecting the right data during testing**
  - GIGO
  - Failure intensity based upon operational profile
    - The number of failures per natural or time unit

Assumptions of Reliability Growth Models

All models have assumptions

- No new errors are introduced by fixes

Use mathematical distribution to represent reliability growth

- Predict failure intensity
- How is this defined? - define failure intensity by the number of failures per natural or time unit (e.g. CPU hour of operation)
- Distribution to represent reliability growth:
  - Poisson
  - Exponential growth

Assumptions of Reliability Growth Models

Use mathematical distribution to represent reliability growth

- SW Physics - are there any underlying aspects to tell us where defects are going
- Collect data points using an operational profile and plug into reliability model
- Produce reliability growth predictions

Problems of Reliability Testing

Operational profile uncertainty

- Is the operational profile an accurate reflection of the real use of the system

High cost of test data generation

- Expensive to generate and check the large number of test cases

Statistical uncertainty for high-reliability systems

- may not be able to generate enough data to create statistically valid conclusions

Problems of Reliability Testing

How do you select a model?

- No universally accepted growth model
- Experiment with different ones, look into assumptions and see by best fit
- No cause and analysis can be used

Students, write your response!

Testing Software Quality Characteristics - 2

When to Stop Testing

Ira A. Fulton Schools of Engineering
Arizona State University

When to stop testing

What criteria do we look into?

- Time to market - early market introduction
- Risk of failures - MTBF
- Cost of continued testing

Trade between three of these criteria

UML-Based Statistical Test Case

Use OP model

- Generate test data based upon OP model
- Random data generated
- Feed into system
- Observe failures rates
- Failures rates fed into models
- Model used to generate reliability of the system

Testing Software Quality Characteristics - 2

Security Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Security Testing

SW correctness versus sw security

Goal

- Confidentiality
  - Data
  - Application
- Integrity - side effects
- Availability

Security Testing

- **Verify access control**
  - Entry
  - Access to functions and data
- **Verify access methods**
  - Cut and paste
  - Screen capture
- **Evaluate malicious input**
- **Evaluate how data is stored and retrieved**
- **Evaluate encryption and data protection**
- **Evaluate decrypted data storage in OS memory**
- **Evaluate data when system is under stress**

Security Testing

- Consider results of component failure (libraries, database, etc)
- Evaluate security when application is denied access to libraries
- Evaluate security when buffers are overflown by long input strings
- Evaluate security when special characters are used as inputs
- Evaluate security when used default usernames/password

Test Management - Part 1

Test Planning

Ira A. Fulton Schools of Engineering
Arizona State University

Test Plan Characteristics

- Well-thought
- Objectives
- Constraints
- Early planning

Test Plan Components

- Testing levels
- Test objectives - functions to test, UAT, security testing, etc
- When to stop testing - meet objectives
- Dependencies - resource, tools, etc
- Assumptions
- Strategies - test data generations, test environments, entry/exit criteria
- Risks

Test Environment

- **Think about**
  - Platform
  - Simulators
  - Tools
- **Based upon objective and strategy**

Test Entry Criteria

What are they?

- Code under configuration management
- Previous test is completed
- No outstanding high priority defects
- Test readiness assessment

Test Schedule

- Identify tasks
- Dependencies of tasks
- Estimate effort and resources needed for tasks
- Assign tasks to individuals or groups
- Map testing tasks to a timeline

Test Risk Management

- Identify risks that impact schedule or effectiveness
- Prioritize risk
- Mitigation plan - include activities to avoid risk
- Develop contingency plans

Test Management - Part 1

Test Schedule

Ira A. Fulton Schools of Engineering
Arizona State University

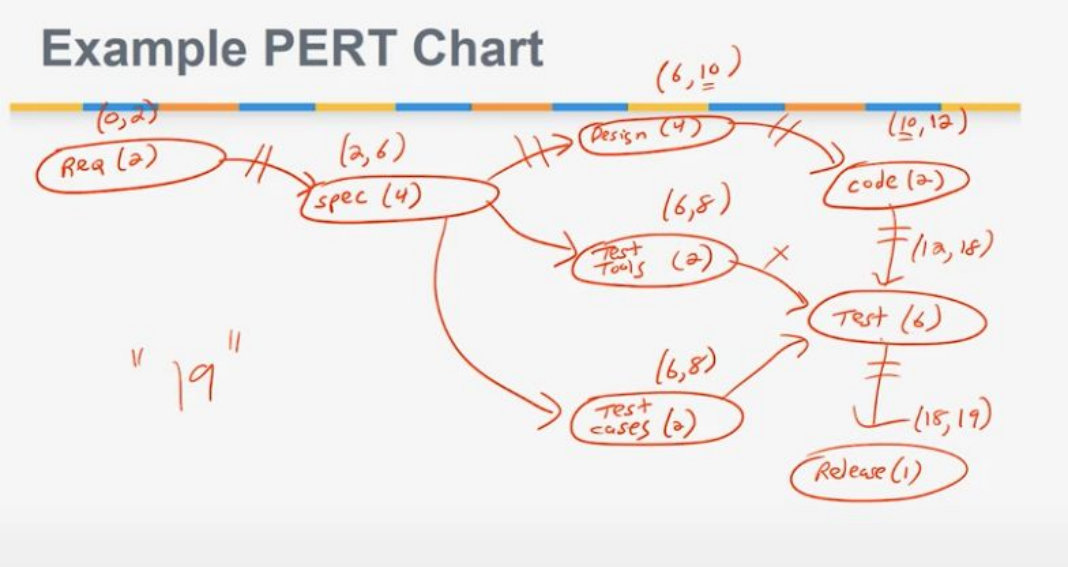
Test Schedule

- Identify tasks
- Dependencies of tasks
- Estimate effort and resources needed for tasks
- Assign tasks to individuals or groups
- Map testing tasks to a timeline

Test Schedule

Analyze Dependencies

- Critical Path - minimum to complete the project



Test Schedule - Assign Task

- Outsource or inhouse
- Similar tasks - same person
- Match knowledge and skill to task
- Train individuals

Test Schedule - Mapped Tasks to Timeline

- **Develop timeline**
- **Buyin and commitment**
- **Gantt chart development**
  - Map to people
  - Start and end date
  - Show parallel tasks
  - Durations
- **Buffers - how long? (10%, 20%) - risk management plan**

Test Management - Part 1

Test Estimation

Ira A. Fulton Schools of Engineering
Arizona State University

Test Estimate

Factors

- Size
- Complexity
- Scope
- Technology
- Desired quality
- Process
- Customer availability
- Quality of code, pass on first attempt

Test Estimate

Strategies

- Historic data
- Cost estimation model - COCOMO, SLIM, SEER-SEM
- As per development effort (⅓)

Test Estimation Issues

- Overestimate versus underestimate
- Misunderstand requirements
- Overlooked tasks
- Time pressure
- Lack of guidelines
- Lack of historic data
- Pressure to reduce estimates

Test Estimation Steps

- **Define estimation responsibilities**
- **Review and clarify objectives, deliverables, milestones and constraints**
- **Define test activities**
  - Understand requirements
  - Training on tools
  - Test cases development
- **Size of product - #requirements, #LOC, #usecases**
- **Estimate size (LOC, test cases, requirements) - topdown, bottom up, 80/20 rule, basis of estimate**
- **Convert size to effort**
- **Check estimates (look at similar products, review by experts)**

Test Management - Part 1

Risk Based Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Risk Based Testing

- **Risk Exposure?**
  - RE = probability of adverse event (error) * consequences
- **Likelihood of failure occurred**
  - Error clusters (due to complexity, changes, development)
  - Severity of failure
  - Different levels of testing
- **Consequences**
  - Understand through developers and customers interaction
  - Issues such as reliability
  - Rigorous failure analysis for complex systems-
    - fault tree
    - Failure mode effect analysis

Risk Based Testing

- **Strategies for prioritization**
  - Test high risk areas tests early
  - Test high risks more thoroughly

Test Management - Part 1

Dealing with Time Crunch

Ira A. Fulton Schools of Engineering
Arizona State University

Time Crunch in Testing

- **What are some of the impacts?**
  - Lack of quality
  - Lack of coverage
  - Poor morale
  - Loss of resources
  - Missed expectations

Time Crunch in Testing

- **Strategies?**
  - Prioritize -
    - based on risks
    - based on criticality
    - Mission needs
    - Stakeholder needs
    - Management directive
    - Experience
    - Sampling - exploratory testing
  - Efficiency - design of experiments

Time Crunch in Testing

- **Strategies?**
  - Prioritize
  - Manage understand tradeoffs
  - Optimize test process more efficiency - Pareto principle (80/20 rule), design of experiments
  - Agile methods
  - Tool support us saving time
  - Know when you are over-testing - reach desired quality

Test Management - Part 1

Improving Test Efficiency through System Test Prioritization

Ira A. Fulton Schools of Engineering
Arizona State University

System Test Prioritization

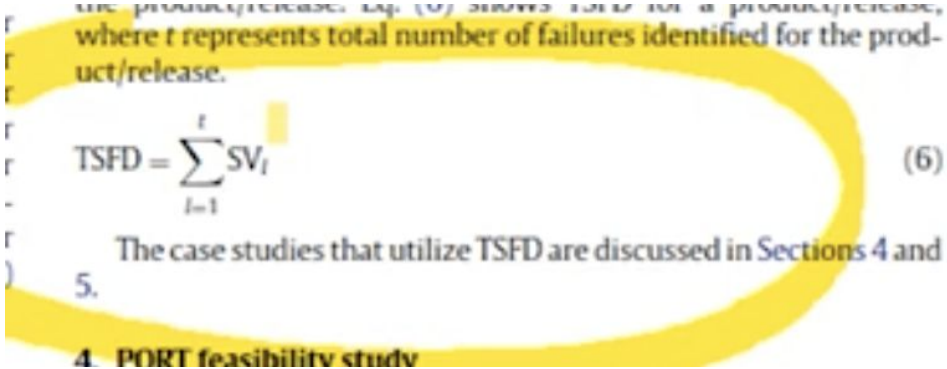
- **Prioritization based on analytics**
- **PORT technique**
  - Find more high severity defects using Port technique compared to random testing

System Test Prioritization

- **PORT technique**
  - T1 - customer assigned priority (1-10)
  - T2 - developer perceived implementation complexity (use complexity measurements - cyclomatic complexity)
  - T3 - fault proneness of the requirements based on history
  - T4 - requirements volatility (how frequently changed)

System Test Prioritization

- **TSFD - total severity of failures detected**
  - How is this calculated
  - Sum of all defects based on severity rating



where t represents total number of failures identified for the product/release.

$$TSFD = \sum_{i=1}^{t} SV_i \quad (6)$$

The case studies that utilize TSFD are discussed in Sections 4 and 5.

4. PORT feasibility study

System Test Prioritization

- TSFD - total severity of failures detected
- Severity levels - 4 levels

follows:

- ***Highly severe (Severity 1)***: Severity 1 is assigned to a failure when a customer can no longer use the product and/or testing must cease until the defect causing the failure is fixed. For Severity 1 failures, we assign a SV of 2^4.
- ***Medium severe (Severity 2)***: Severity 2 is assigned to a failure when there is a work-around for the failure and the product can be used with the work around. For Severity 2 failures, we assign a SV of 2^3.
- ***Less severe (Severity 3)***: Severity 3 is assigned to a failure for which a fix can be done in later versions. For Severity 3 failures, we assign a SV of 2^2.
- ***Least severe (Severity 4)***: Severity 4 is assigned to a failure for which a fix may be done at later versions or not done at all. A SV of 2^1 is assigned to failures with Severity 4.

Test Management - Part 1

Risk Driven Model-Based Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Risk-Driven Model-based testing

- Prioritize based on usage (OP)
- Fault likelihood

Test Management - Part 1

Test Exit Criteria

Ira A. Fulton Schools of Engineering
Arizona State University

Test Exit Criteria

- Run out of time
- Coverage
- Reliability model
- Analytics
- Found all defects
- Customer satisfied
- Test objectives

Test Exit Criteria

- **Defect density**
  - #of defects per lines of codes
- **Defect pooling**
  - Compare defects found
    - Uniques defects
    - Estimated total defects
    - Estimate Remaining defects
- **Defect seeding**
  - Found defects versus inserted defects

Test Exit Criteria

- **Trend Analysis - increasing reliability**
  - Mean time to failure
  - Cumulative number of failures
  - Number of failure per unit of time (#failure/hr)

Test Management - Part 1

Test Documentation

Ira A. Fulton Schools of Engineering
Arizona State University

Test Documentation

- **Templates for:**
  - Test plan
  - Test case
  - Test incident report
  - Test summary report
  - Others

Test Documentation

- **Defect priority versus severity?**
  - Priority - project consideration
  - Severity - impact to customer

Test Documentation

- **Review of defects**
  - Completeness
  - Repeatability
  - Clarity
  - Severity evaluation

Test Documentation

- **Test Summary Report**
  - Summary of what was tests
  - Variances - extra, or not run
  - Assessment - traceability
  - Summary of results
  - Evaluation (prediction)
- **Formality and audience**
- **Deliverable or not**
- **Support regression testing**

Test Documentation

- **Test Summary Report**
  - Summary of what was tests
  - Variances - extra, or not run
  - Assessment - traceability
  - Summary of results
  - Evaluation (prediction)
- **Formality and audience**
- **Deliverable or not**
- **Support regression testing**

Test Management - Part 2

Test Tracking

Ira A. Fulton Schools of Engineering
Arizona State University

Test Tracking

- **What are we tracking?**
  - Test Progress
    - % test developed
    - % test executed
    - % requirements tested
      - Testing schedule and effort progress assessed via earned values
  - Product quality (test entry and exit criteria)

Earned Value

- **Earned values are a technique for tracking**
  - Schedule
  - Cost
- **Track both schedule and cost progress**
  - **Budgeted Cost of Work Scheduled (BCWS)**
  - **Budgeted Cost of Work Performed (BCWP)**
  - **Actual Cost of Work Performed (ACWP)**

Earned Value Example

Tasks	EV	Tasks	EV
1A	50	2A	30
1B	40	2B	40
1C	30	2C	30
1D	20	2D	50
1E	50	2E	40
1F	30		

	Week 1	Week 2	Week 3	Week 4
BCWS	90	210	370	410
BCWP	120	220		
ACWP	100	170		

Week 1	Week 2	Week 3	Week 4
1A, 1B	1C, 1D	1E, 1F	
	2A, 2B	2C, 2D	2E

Week	Work Completed	Cost
1	1A, 1B, 1C	100
2	1D, 2A, 1E	70

- Based on earned values the project is ahead of schedule at the end of week 2
  - BCWS = 210
  - BCWP = 220
- Based on earned values the project is below budget at the end of week 2
  - BCWP = 220
  - ACWP = 170

Test Management - Part 2

Reading - Test Management

Ira A. Fulton Schools of Engineering
Arizona State University

Test Management - Case Study

- **Test Effectiveness or Defect Removal Efficiency (DRE) -**
  - Productive with minimum waste or effort
  - **Test Effectiveness = Defect found / defects to be found**
  - Example:
    - 90 bugs are found
    - 10 bugs are found by user
    - Test effectiveness = 90 / (90+10) = 0.90

Test Management - Case Study

- **Test Efficiency - having a definite or desired effect**
  - $/test case
  - How to improve efficiency?
    - Automation help improve efficiency

Test Management - Case Study

- **Pass rate ?**
  - ratio of tests that passed to the total number of tests
- **Passes versus failures**
  - Ratio of tests that pass to the number of tests fail

Test Management - Case Study

- **POFA?**
  - Number of tests pass on first attempt
- **POFA comparison**
  - 90%
    - Less time spend for retesting, documenting defects and waiting for defects to be fixed.
  - 50%

Test Management - Case Study

- **Test progress S- curve?**
  - attempted, successful and planned test cases
  - Number of planned test starts low then increases and finally levels of as it gets closed to release time

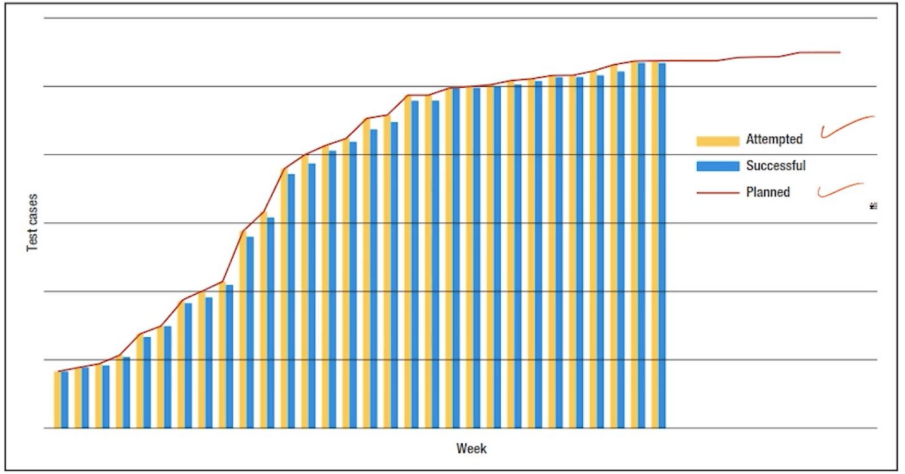


FIGURE 1. Sample test process S curve.[3] Cumulative graph of attempted, successful, and planned test cases over time.

Test Management - Part 2

Test Process Improvement

Ira A. Fulton Schools of Engineering
Arizona State University

Test Process Improvement (cont)

- **Phases:**
  - GQM paradigm (goal-question-metrics)
    - Define goals of the measurement process
    - Derive the questions that must be answered to meet the goals
    - Develop metrics to answer the questions
  - Example:
    - Goal: reduce testing time or find more severe defects
    - Questions: where time is spent,
    - Metrics: ?

Test Process Improvement (cont)

- **Phases:**
  - Process Redesign
    - Explore ways of eliminating or combining activities
    - Explore ways of eliminate rework
    - Explore ways of reduce task variance

Test Process Improvement (cont)

- **Phases:**
  - Implement Process Improvement
    - Begin with candidates that are well defined
    - Pilot candidates
    - Measure progress

Test Process Improvement Tools

- **Post-Mortem/Lessons Learnt/Retrospectives**
  - Document lessons learned
  - Interview key personnel
  - Statistical analysis
  - Investigate major problems
  - Identification of what went well and what didn’t go well

Test Process Improvement Tools

Lessons Learnt Topics

- Overall schedule analysis
- Adequacy of entrance criteria
- Overall quality of testing
- Team interaction
- Effectiveness of communication and collaboration with development
- Effectiveness of test estimates
- Issues with test environment
- Defects missed
- Areas where time was wasted

Test Management - Part 2

Test Outsourcing

Ira A. Fulton Schools of Engineering
Arizona State University

Test Outsourcing

- **When and how to outsource**
- **Factors:**
  - Cost - is it cheaper or not?
  - Time to complete
  - Tools
  - Quality
  - Advanced technology
  - Strategic value of system
  - Strategic alliance
  - Speed of development - is it faster or not?
  - Desire for level staffing (level up or down)

Test Outsourcing

- **How to outsource**
  - **Define work**
    - Maximize effectiveness
    - Match skills and capabilities
    - Minimize communication
    - Minimize dependencies
    - Minimize risk with knowledge transfer

Test Outsourcing

- **How to outsource**
  - **Develop SOW**
    - Identify all tasks to be performed
    - Identify process to be followed - how?
    - Identify maintenance responsibilities

Test Outsourcing

- **How to outsource**
  - **Risk management**
    - Identify risks
    - Mitigation plan to reduce risk
- **Estimate resource needed (in house versus contractor)**
  - Estimate supplier test effort
  - Estimate effort for vendor

Test Outsourcing

How to outsource

- How to select
  - Strategy vendors
  - Legal terms of contract
  - Negotiating contract
- Subcontract management
  - Conveying and explaining requirements
  - Monitor - need visibility
  - Resolve problems

Test Outsourcing

- **How to outsource**
  - Factors in selecting
    - Staffing
    - Strategic alliance
    - Location
    - Capabilities
    - Cost
    - Business viability
    - Similarity of tools/processes
    - Domain expertise
    - Prior performance on similar work

Test Outsourcing

- **How to outsource**
  - Tracking and Oversight
    - Communication
    - Visibility
    - Metrics
    - Reviews
    - Risk management
    - Escalation procedures
    - Approval of invoices

Test Outsourcing

How to outsource

- Acceptance of Work
  - Formal procedure
  - Tests developed
  - Tests run
  - Test through or complete?
    - Code coverage
    - Requirement traceability
    - Mutation test

Test Management - Part 2

People Management

Ira A. Fulton Schools of Engineering
Arizona State University

People Management

- **Improve - 3 factor**
  - Process - CMMI, ISO
  - Technology - Case
  - People - Team player

People Management

- **Effective use of people**
  - Test Lead
    - Negotiators (schedule, resource, entry criteria)
    - Establish priorities
    - Manage team
      - Leadership, delegation, communication, motivation, negotiation, problem solving

Test Management - Part 2

Test Inspection

Ira A. Fulton Schools of Engineering
Arizona State University

Test Inspection

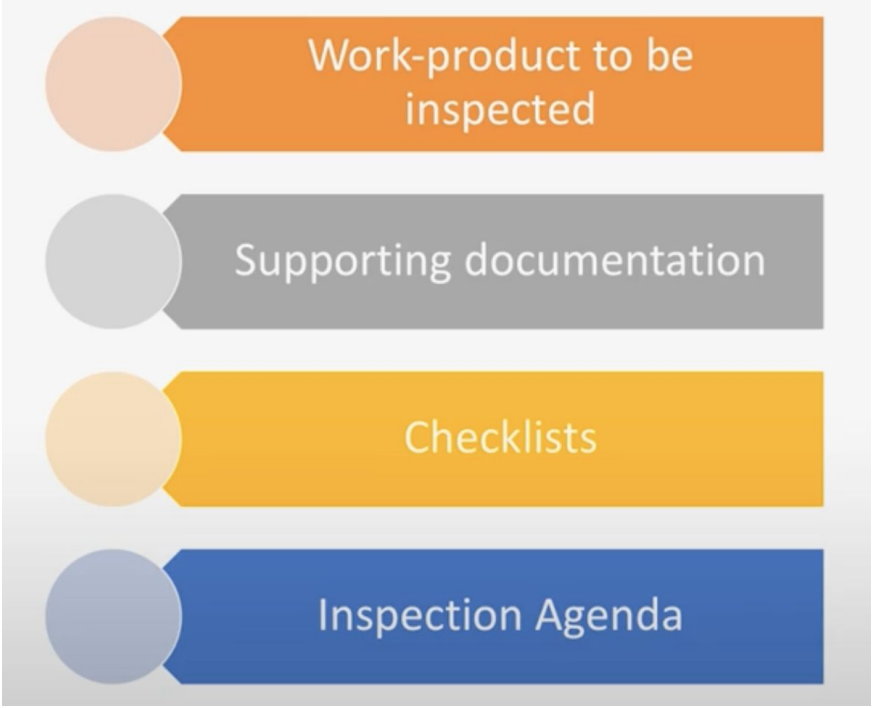
- **Review versus Inspection?**
  - Formal statistical process control method
  - Advanced preparation
  - Utilize checklists and rules
  - Metrics gathered
  - Analysis of metrics

Test Inspection

- **What is inspected during testing?**
  - Test plan
  - Test cases
  - Test incident reports
- **What other work products needed?**
  - Additional docs like requirements
  - Checklists (job aid, customized)
  - Inspection agenda

Test Inspection

Inspection package



Test Inspection

- **Creating checklist**
  - Best practice
  - Experience
- **Requirements checklist**
  - Testability
  - Completeness
  - Correct
  - Consistent
  - Clarity

Test Management - Part 2

Causal Analysis

Ira A. Fulton Schools of Engineering
Arizona State University

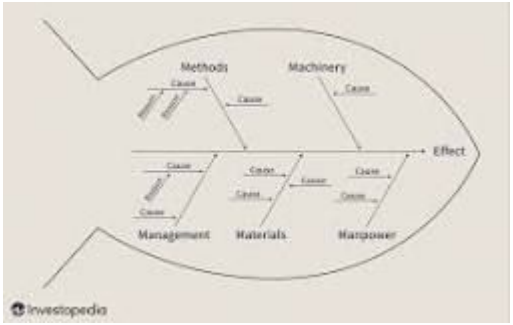
Causal Analysis

- **Process improvement technique introduced by IBM in 1983**
  - Collect defect data
  - Analyze of possible causes
  - Identify common causes
  - Identify possible solutions

Causal Analysis

Ishikawa Diagram - Fishbone diagram

- Causes of a defect
- Some causes of defects are
  - Communication failure
  - Oversight
  - Education
  - Transcription



Causal Analysis

5 Whys

- A technique for finding root cause of a defect by asking 5 times the question why?

Causal Analysis

- **Strategies for developing possible solutions**
  - Group defect and direct the high visibility/high impact defects
  - Oversight problems
    - automation
  - Education problems
  - Communication problems
  - Transcription problems
    - Reviews, automation

Test Management - Part 2

Causal Analysis Paper

Ira A. Fulton Schools of Engineering
Arizona State University

Causal Analysis

- **Who does RCA?**
  - Developer and testers

- **ODC - Orthogonal defect classification (defect analytics)**

Causal Analysis

TABLE I. ROOT CAUSES FOR MAIN DEFECT TRIGGERS

Root Cause	Defect Trigger
Lack of traceability verification culture	Traceability/Compatibility
Lack or inefficient usage of tools that support traceability across lifecycle phases	
Lack of appropriate test planning and test strategy definition	Test Coverage
Lack or inefficient testing tool and testing environment support	
Incomplete tests specification and execution	
Review process related root causes	Document Consistency/Completeness (Internal Document)
Documentation related root causes	Same as above
Deficient usage of tools and applicable processes	Same as above
Unclear or missing/incomplete specifications	Same as above Logic / Flow
Ambiguous or unclear architecture definition	Logic / Flow
Lack of usage of tools that support data and control flow analysis	Logic / Flow
Inappropriate architecture support tools or tool usage	Design Conformance
Deficient specification or design artefacts	Design Conformance

TABLE II. RECOMMENDED V&V MEASURES

Define appropriate test plans and strategies, especially unit and integration tests. The soundness of the test plans and strategies will reflect in the success of the validation; [Test Coverage]
Ensure appropriate (or automated) traceability analysis at every stage of the development lifecycle; [Traceability/Compatibility]
Improve testing completeness, coverage and reviews; [Test Coverage]
Implement non-functional tests (fault detection, fault injection, redundancy, etc.); [Test Coverage]
Apply/develop tools to validate the implementation and design compliance. [Document Consistency/Completeness; Logic/Flow; Design Conformance]

Test Management - Part 2

Test Maturity Model

Ira A. Fulton Schools of Engineering
Arizona State University

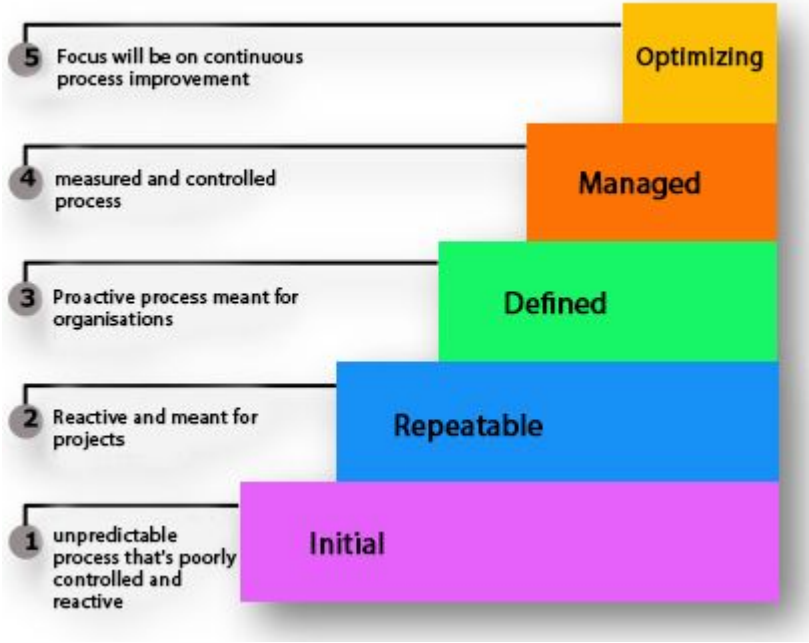
Test Maturity Model

- **CMMI versus TMMI**
- **Why needed it?**
  - Used to select a company and assess their maturity

Students, write your response!

CMMI

- CMMI versus TMMI - 5 levels
- 1-Initial
- 2-Repeatable - expertise in individuals
- 3-Defined - processes defined and documented
- 4-Managed - metrics are used to guide process
- 5-Optimizing- continuous improvement



TMMI

- TMMI - 5 levels
- 1-Initial - ad hoc, no tracking
- 2-Phase definition - test phases planned, testing method (EV)
- 3-Integration - dev process, monitor and track test process, training, risk management
- 4-Managed - metrics are used to guide test process (effectiveness
- 5-Optimizing- continuous improvement, causal analysis, prevent defect reaching user

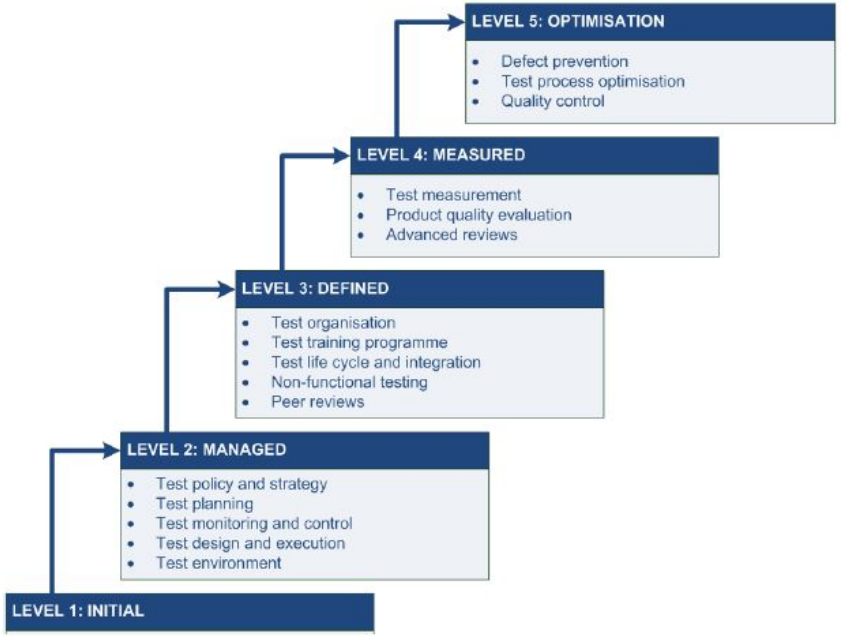


Figure 1 - TMMi model.

Dynamic Testing

- **Definition of testing types**
  - Performance, volume, security, load, stress, etc
- **Things to consider in different testing types**
  - Configuration testing
- **Availability - Reliability**

Test Planning

- Calculate the critical path in PERT chart
- Impact on critical path
- Assessment of an impact to a schedule based on delay in an activity
- Risk-based testing
- Estimation of test effort
- Reliability growth model
- Outsource - in house decision

Test Documentation

- Testing SOW content
- Testing outsourcing SOW
- Test plan doc
- Test case doc
- Test case incidents

Test Cycle Review

- Calculation of Failure Intensity
- Trends in sw and hardware failure intensity
- Assessment of how project is doing in terms of schedule and budget
- Assessment of test effectiveness
- Post - mortem
- GQM

Test Management

- People Management
- Team Structure