Test Management Part 1

Risk Based Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Perform risk based testing

Risk-Based Testing Strategy

- **Applicable when project constraints make it necessary to prioritize testing**
- **High risk areas are identified as a function of the:**
  - Likelihood of a failure occurring
  - Severity of failure should it occur
- **Can be applied at various levels of abstraction:**
  - Subsystem
  - Feature
  - Component

Assessing the Likelihood of a Failure

- **Errors cluster**
- **Some areas of the system may be more error-prone due to:**
  - Complexity
  - New or changed code
  - Outsourced development
  - Poor history

Assessing the Consequences of a Software Failure

- Requires interaction with customers and developers
- Failures of capabilities can be assessed in terms of severity
- Severity must address issues such as system:
  - Reliability
  - Availability
  - Performance
  - Usability
  - Compatibility
  - Maintainability
- Requirements / use-cases should be prioritized to support risk-based testing

Assessing the Consequences of a Software Failure

- For more complex systems, consequences of a component or function failure may not be obvious in terms of its severity
- Rigorous failure analysis can be performed with assistance of developers using techniques such as:
  - Fault trees
  - Failure mode effect analysis

Risk-Based Testing Strategy

Summary

Test Management Part 1

Test Documentation

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Identify the different types of test documentation

IEEE Test Documents

See IEEE STD-829 IEEE Standard for Software Test Documentation

Templates are provided for:

- Test plan
- Test case
- Test Incident Report
- Test Summary Report
- Other test documentation

Test Case

- Test case specification identifier
- Test items
  - References to documents used to produce test
- Input specifications
- Output specifications
- Environment for executing tests
- Special test procedures to follow
- Intercase dependencies

Test Incident Report

Test incident report identifier

Summary

Incident description

- Inputs
- Expected results
- Actual results
- Date and time
- Environment
- Attempts to repeat

Impact

Test Incident Reports

- **Must be carefully written**
- **Target audience includes:**
  - Programmers
  - Other testers
  - Managers
  - Quality Assurance
  - CCC Board

- **Must provide enough information for each viewer to do their job**

Developer Mentality

Testers must understand what it is like to work as a programmer

- Complex tasks
- Vague and changing requirements

Most developers are specialized in a part of the system

- Testers are generalists
- Testers must help developers understand the big picture

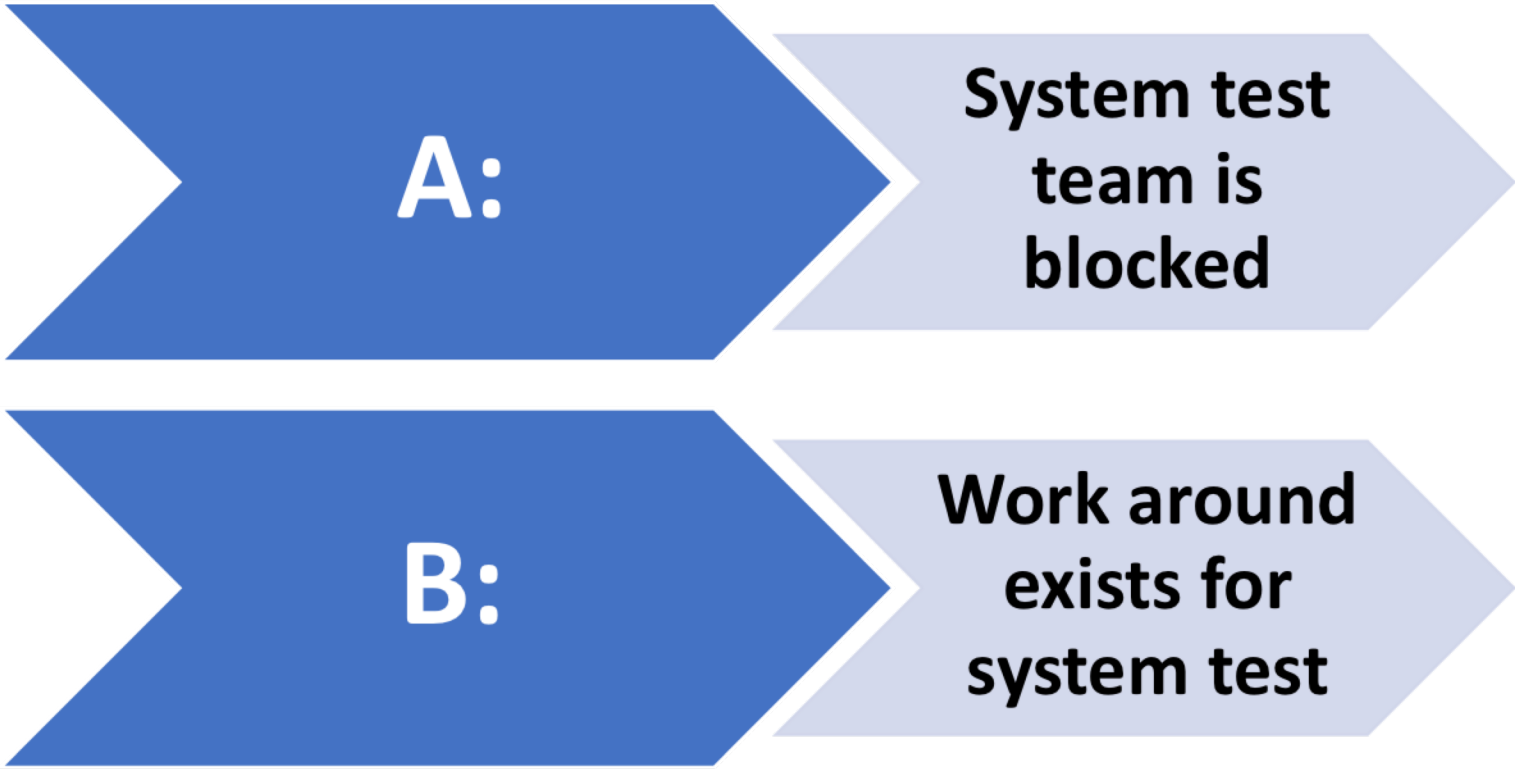
"context"

Severity vs Priority

- Testers need to recognize the difference between severity and priority
- Severity reflects the customer impact of the error
- Priority reflects project considerations

- Not all high severity errors are high priority
- Not all high priority errors are high severity

System Test Problem Priorities



Review Defect Reports?

To save time, some organizations have bug reports reviewed by another tester

Review criteria include:

- Completeness
- Repeatability
- Clarity
- Severity evaluation

Test Summary Report

Test Documentation

Agile development methods have forced rethinking of documentation requirements

Level and formality of test documentation must be based on:

- How will documentation support the testing activity
- Is documentation a deliverable
- Are their regulatory concerns
- Will documentation support tracking activities
- Support for regression testing

Summary

Test Management Part 1

Test Estimation Steps

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Follow a process
for estimating
testing efforts

Importance of Estimates

Estimates are the foundation for:

- Testing budget
- Allocation of staff and resources

Testing estimates are often made as a part of the overall development project estimate

Major Causes of Inaccurate Estimates

- Misunderstanding of requirements
- Overlooked tasks
- Insufficient analysis when developing estimates due to time pressure
- Lack of guidelines for estimating
- Lack of historical data
- Pressure to reduce estimates

Generic Estimation Process

1. Determine estimation responsibilities
2. Review and clarify testing objectives, deliverables, milestones and constraints
3. Identify testing tasks
4. Select appropriate size measure for testing work
5. Select size estimation method
6. Estimate and document size
7. Estimate and document effort

Identification of Testing Tasks

- **A set of testing activities must be defined**
- **Testing tasks include:**
  - Understanding requirements
  - Training on tools
  - Test case development

Select Appropriate Size Measure

Size of the testing activity must be estimated utilizing some type of standard component

Possible testing size measures include:

- Number of requirements
- Number of use cases / scenarios
- Lines of code to be tested

Select Size Estimation Method

Top-Down / Analogy

Bottom-up

Top-Down / Analogy

- An expert develops an estimate based on past similar projects
- Does not work well for new types of projects
- May fail if estimator does not take into account differences between projects such as expertise, difficulty of testing, stability of requirements, etc.

Bottom-Up Estimation

- **Breaks the testing effort into parts**
- **Parts may consist of:**
  - Functions to test
  - Types of testing to perform
- **Each part is estimated separately and the parts are summed to create the estimate.**
- **A work breakdown structure may be developed to organize the parts**

Estimate and Document Size

- 80/20 rule applies to size estimation
- Accuracy of the size estimate must also be communicated
- Prior to effort estimation, the size of work must be estimated and documented along with the basis of the estimate and the assumptions made

Estimate and Document Project Effort

- **A size unit for effort must be selected**
  - Staff hours
  - Staff months
- **actual effort is a function of:**
  - Size estimates
  - Estimated productivity

- **Effort estimate must be documented along with the basis for the estimate and the assumptions made**

Strategies for Checking Estimates

- Check that estimates for similar parts of the system are the same
- Check resource estimates of parts of your system with similar parts of other completed systems
- Review with other experts

Summary

Test Management Part 1

Test Estimation

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Estimate testing effort

Estimating System Testing Time

House Cleaning Analogy

Developing Estimates

Once the variables to take into account in developing the testing estimate have been identified, an estimate based on them must be produced

Approaches for developing estimates based on variables include:

- Utilization of historical data
- Calculation via a cost estimation model
- Generating a test estimate based on a percentage of development estimate

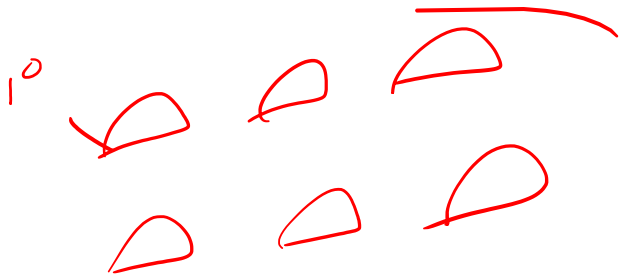
Summary

Test Management Part 1

Test Exit Criteria

Ira A. Fulton Schools of Engineering
Arizona State University

Objective



Objective

Select test exit criteria

When to stop testing?

1. Out of time / $
2. coverage
3. Reliability models
4. curves / graphs
5. analytics found all defects
6. customer is satisfied
7. test objectives

System Test Exit Criteria

The best criteria for stopping testing is when the test objectives have been met

Typical approaches for determining that the system is ready for release include:

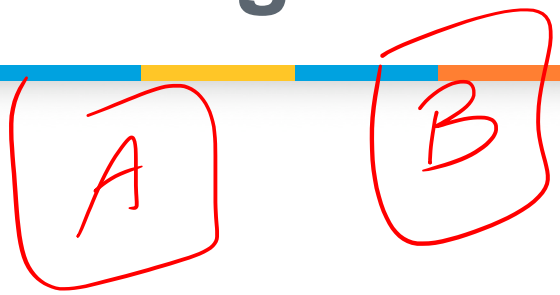
- Measuring defect density
- Defect pooling
- Defect seeding
- Trend analysis
- Reliability modeling

Measuring Defect Density

- **Defect density is defined as number of defects per thousand lines of code**
- **If available, historical defect density data might be used to predict expected number of defects to be found in system test**
- **Actual number might be compared against expected number to determine if system is ready for release**

- **Problems with this approach include:**
  - Lack of historical data
  - Defect injection rate for current system might be different than historical data
  - Defect containment for previous phases might be different than historical data
  - Defects found may not impact customer's perception of the system

Defect Pooling



- Appropriate to apply when operational usage scenarios are being executed
- Requires defect reports to be broken into 2 groups which can be tracked separately
- Assumes that each group of defects reflects independent testing of whole system based on operational usage
- Can be implemented many ways such as:
  - Splitting the system testers into 2 groups
  - Collecting defect data from 2 independent beta test groups

Defect Pooling Approach

Calculate the following assuming:

- $Defects_A$: is the number of defects found by Group A
- $Defects_B$: is the number of defects found by Group B
- $Defects_{A+B}$ is the number of defects found by both Group A and B

Unique Defects =
$(Defects_A + Defects_B) - Defects_{A+B}$

Estimated Total Defects =
$(Defects_A \times Defects_B) / Defects_{A+B}$

Estimated Remaining Defects =
Estimated Total Defects - Unique Defects

Defect Pooling Example

Assume:

- Group A found 30 defects
- Group B found 40 defects
- 20 defects were common to both Group A and Group B

$Defects_A = 30$ $Defects_B = 40$ $Defects_{A+B} = 20$

Unique Defects = (30 + 40) - 20 = 50

Estimated Total Defects = (30 x 40) / 20 = 60

Estimated Remaining Defects = 60 - 50 = 10

Defect Seeding

- Controversial approach involving "seeding" defects into the system
- Assumes that defects can be inserted into the system that are representative of defects that customers will encounter
- Assumes ability to detect remaining defects is equivalent to ability to detect seeded defects
  - Estimated Total Defects = (seeded defects planted / seeded defects found) x (normal defects found)

Defect Seeding Example

Assume:

- 20 defects are seeded
- 10 seeded defects are found by test
- 50 additional defects are found by test

Estimated Total Defects = (20 / 10) x 50 = 100

Estimated Remaining Defects = 100 - 50 = 50

Trend Analysis

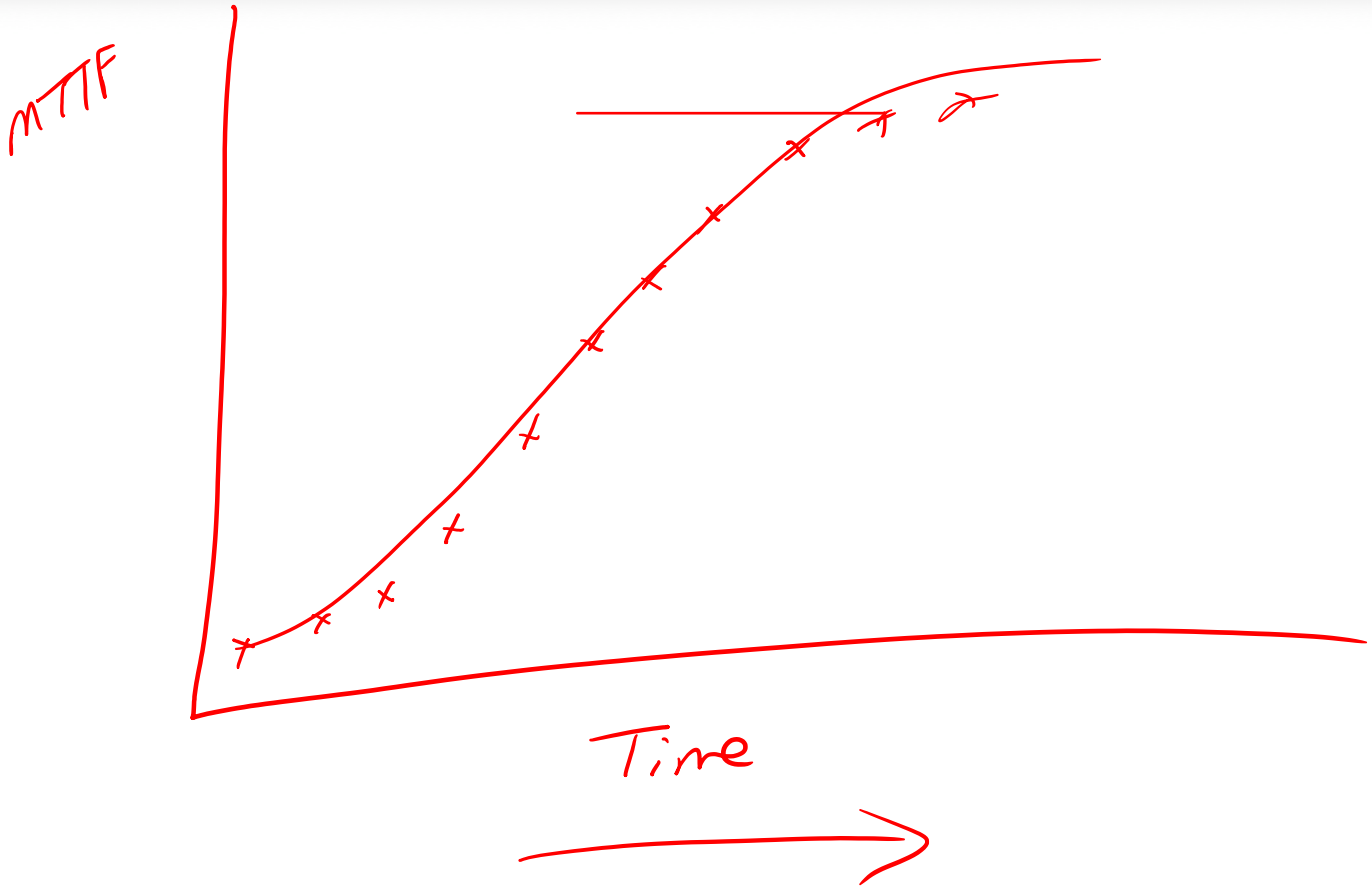
Software readiness may be assessed via analysis of trend data:

- Time to failure
- Cumulative number of failures
- Number of failures per unit of time (failure intensity)

One of three trends can be identified:

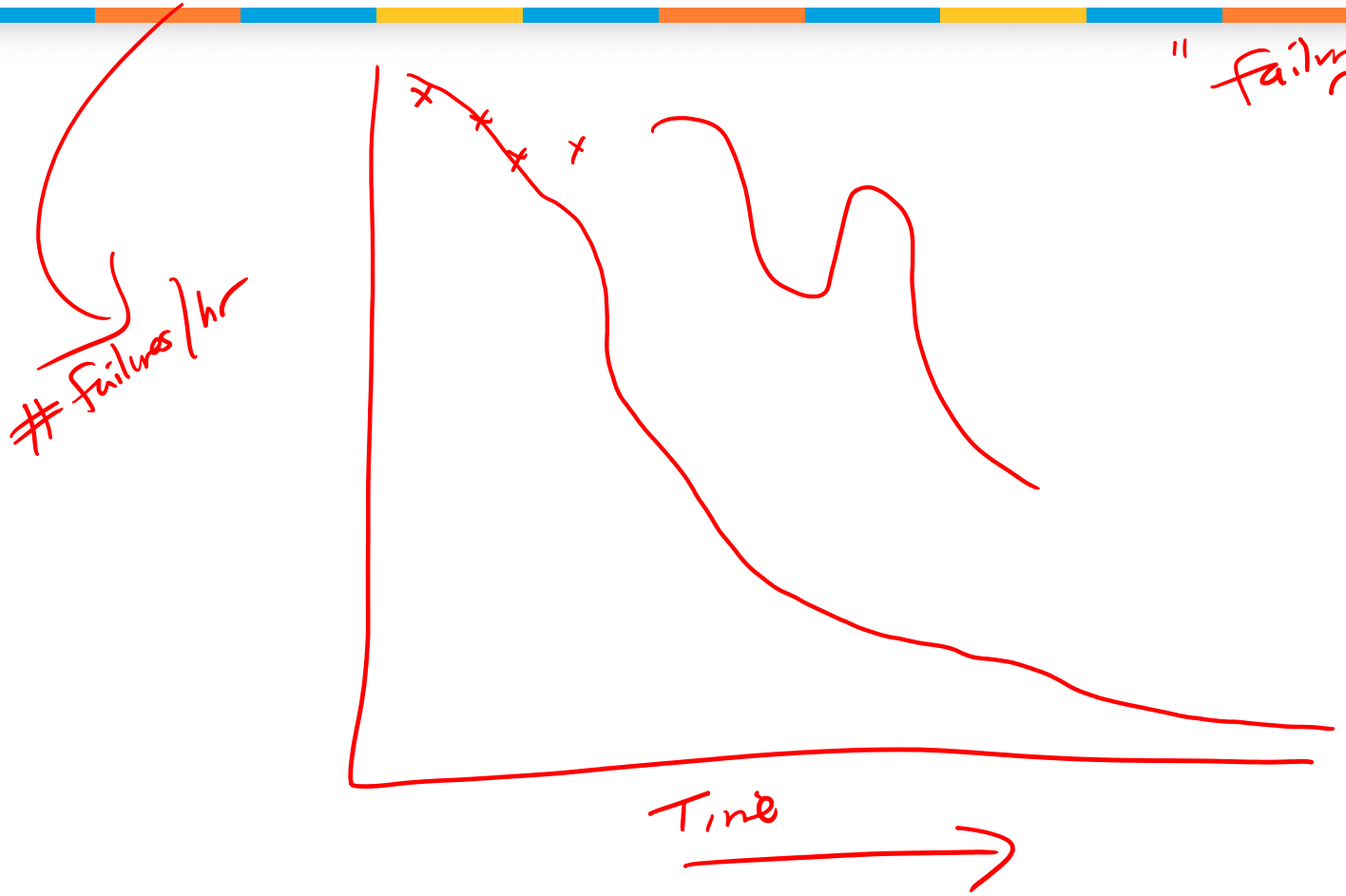
- Decreasing reliability
- Increasing reliability
- Stable reliability

Time to Failure



Cumulative Number of Failures

Number of Failures Per Unit of Time



Trend Analysis (Decreasing Reliability)

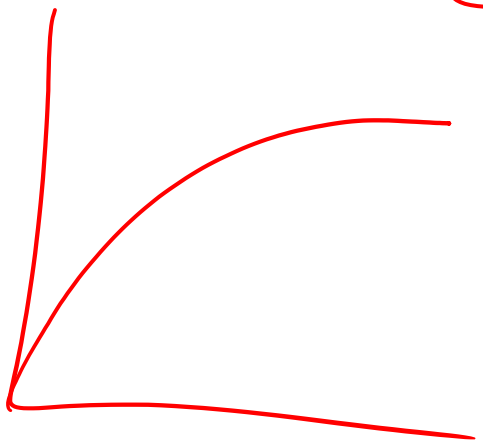
Expected at the start of a new activity

- New testing phase
- New type of testing
- Different user profile

Long duration signals significant software problems

Trend Analysis (Increasing Reliability)

- Normally good news



- Sudden increase may, however, be due to:
  - Changing test effort
  - Test burnout
  - Unrecorded failures

Summary

Test Management Part 1

Test Planning Overview

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Identify the major components of a system test plan

Test Planning

Test plans should be written for all testing levels:

- Unit
- Integration
- System
- Beta
- Acceptance

System Test Plan

- A well-thought out system test plan is essential to success of a testing effort
- System test plan must reflect an in-depth understanding of the objective of the system test as well as project constraints
- System test planning must begin early (during the software development requirements phase)

System Test Plan (continued)

The system test plan must address:

- System test objectives
- Dependencies and assumptions
- Adopted test strategy
- Specification of the test environment
- Specification of system test entry and exit criteria
- Schedule
- Risk management

System Test Objectives

- The system test plan must clearly define the objectives of the system test activity
- Possible objectives were presented in Unit 1

Dependencies and Assumptions

When creating the system test plan, all dependencies and assumptions must be identified

Examples include:

- Resource availability
- Software completed on time
- _____
- _____

Testing Strategy

Testing strategy defines how testing objectives will be met within project constraints

Testing strategy determines:

- Techniques to be used for test data generation
- Test environment
- Entry and exit criteria
- Schedule

Testing strategy is often risk-based

Specification of the Test Environment

Test environment issues include:

- Platforms to test on
- Simulators
- Testing tools

Selection of test environment is based on objective of testing and test strategy chosen

- Performance testing objective may require load generation tools
- Configuration testing objective may require additional resources and/or simulation tools

System Test Entry Criteria

- **Established based on test strategy to maximize test effectiveness**
- **Problems with beginning system test too early include:**
  - Inability to run all tests
  - Excessive communication with developers on problem fixes
  - High degree of retest
- **Possible entry criteria include :**
  - Code under configuration management
  - Completion of integration test
  - No outstanding high priority problems
  - Successful completion of system test readiness assessment

System Test Readiness Assessment

- Developed early in the project in conjunction with development
- Identifies functions and code stability needed to effectively begin system test
- Provides a concrete entry criteria for system test
- Provides a way for development to prioritize their activities as the start of system test grows near

Creation of System Test Schedule

Creation of a testing schedule requires the following activities:

- Identify all of the testing tasks to be performed
- Identify dependencies among the testing tasks
- Estimate the effort and resources needed to perform each task
- Assign tasks to individuals or groups
- Map testing tasks to a time line

Test Plan Risk Management

- System testing risks correspond to scenarios that could impact testing schedule or effectiveness
- Testing risks can be identified via checklists or previous project "lessons learned”
- Testing risks must be prioritized and mitigated
- Prioritization is based on likelihood of risk occurring and consequences
- Risk mitigation involves reducing the likelihood of the risk occurring and/or developing contingency plans to minimize impact of the risk should it occur

Summary

Test Management Part 1

Test Schedule

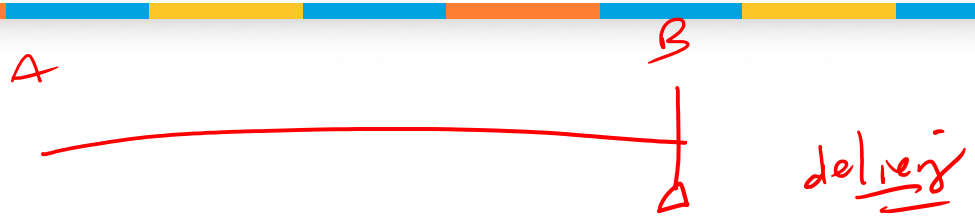
Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Create a test schedule

Creation of System Test Schedule



Creation of a testing schedule requires the following activities:

- Identify all of the testing tasks to be performed
- Identify dependencies among the testing tasks
- Estimate the effort and resources needed to perform each task
- Assign tasks to individuals or groups
- Map testing tasks to a time line

Testing Tasks Examples

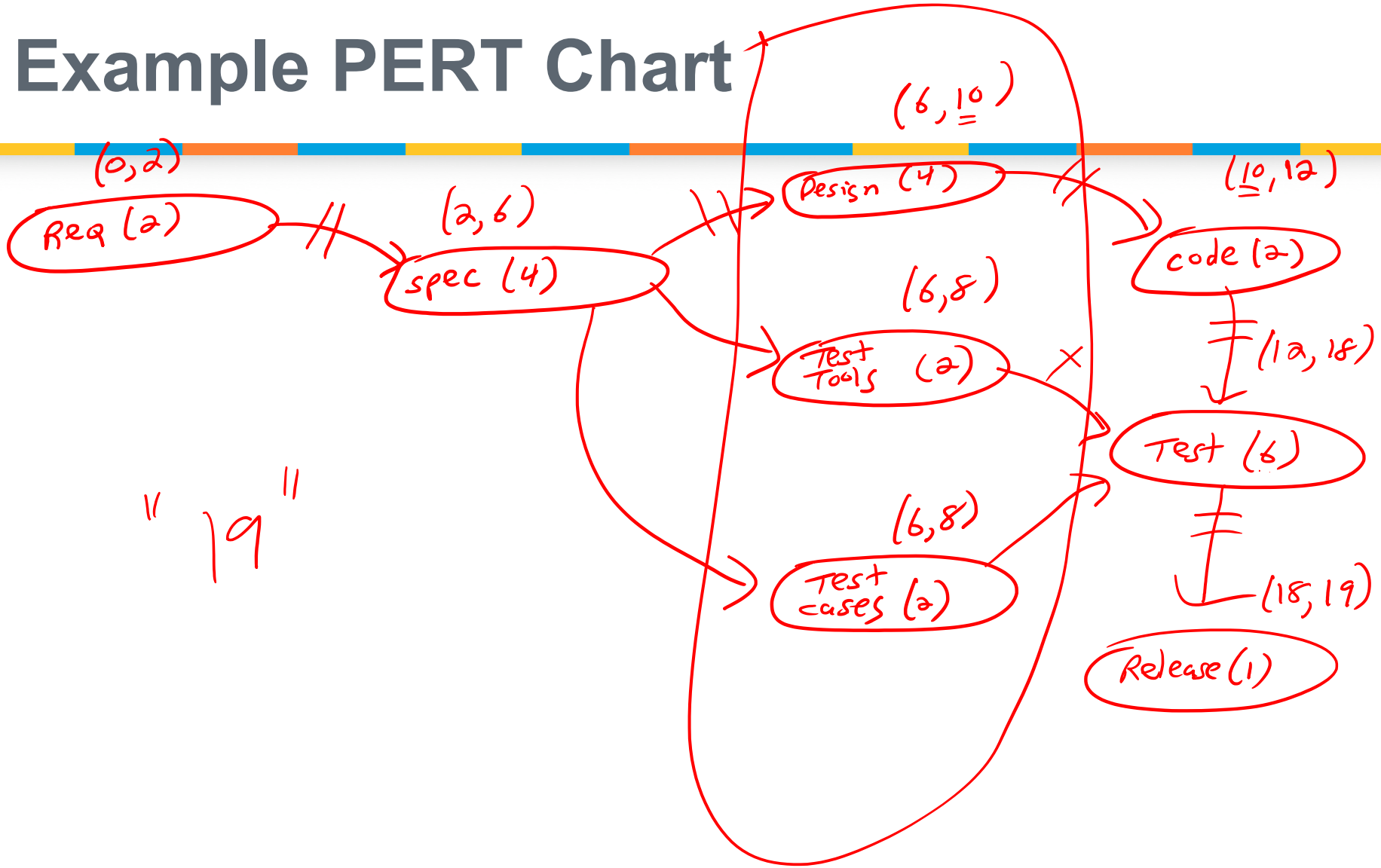
- Develop test plan
- Understand requirements
- Develop tests
- Review tests
- Install test environment
- _____
- _____
- _____

Analyze Dependencies

- After tasks are identified, a dependency analysis among tasks must be performed
- Dependencies can be documented in a PERT Chart
- PERT (Program Evaluation and Review Technique) was developed by the Navy in the 1950's

- Chart is a network whose nodes represent project activities and their associated duration's (often calendar weeks) and whose links represent precedence relations

Example PERT Chart



Critical Path Analysis

- What is the minimum time it will take to complete the project?
- What activities are critical to being able to complete the project in minimum time?
- What activities can be done in parallel?
- How long can each activity be delayed before it affects the finish date?

Critical Path Identification

- Critical path(s) is the path through the PERT chart with no slack time
- Can be identified by associating with each node, its earliest start and finish time
- Those paths where the earliest start time is always equal to the predecessor's nodes earliest finish time correspond to critical paths

Estimating System Testing Time

- A critical part of test planning is estimating the time needed to meet the testing objectives
- Overestimates lead to inefficient testing and delayed product release
- Underestimates lead to lots of overtime, high stress and probable ineffective testing

Assign Task Responsibilities

- Assign similar tasks to the same person
- Minimize necessary communication
- Match knowledge and skills to the task

- Assign tasks to people so that they learn and grow
- Attempt to accommodate preferences

Map Tasks to a Time Line

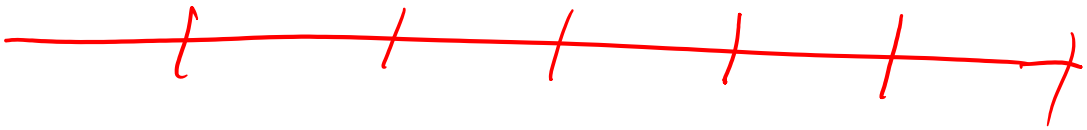
Schedule must be well thought out and take into account:

- Constraints
- Task dependencies
- Availability of personnel
- Risks

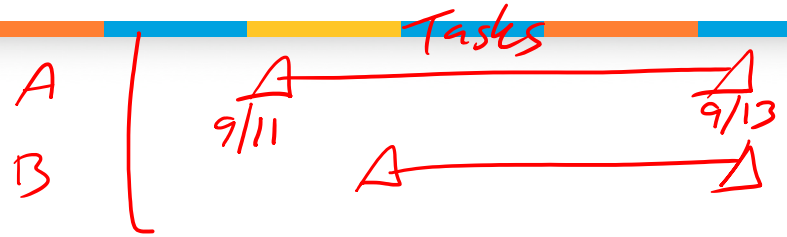
Developing a schedule is an iterative process:

- Adjust tasks, durations, resources and sequencing

Participants must commit to the schedule



Gantt Chart



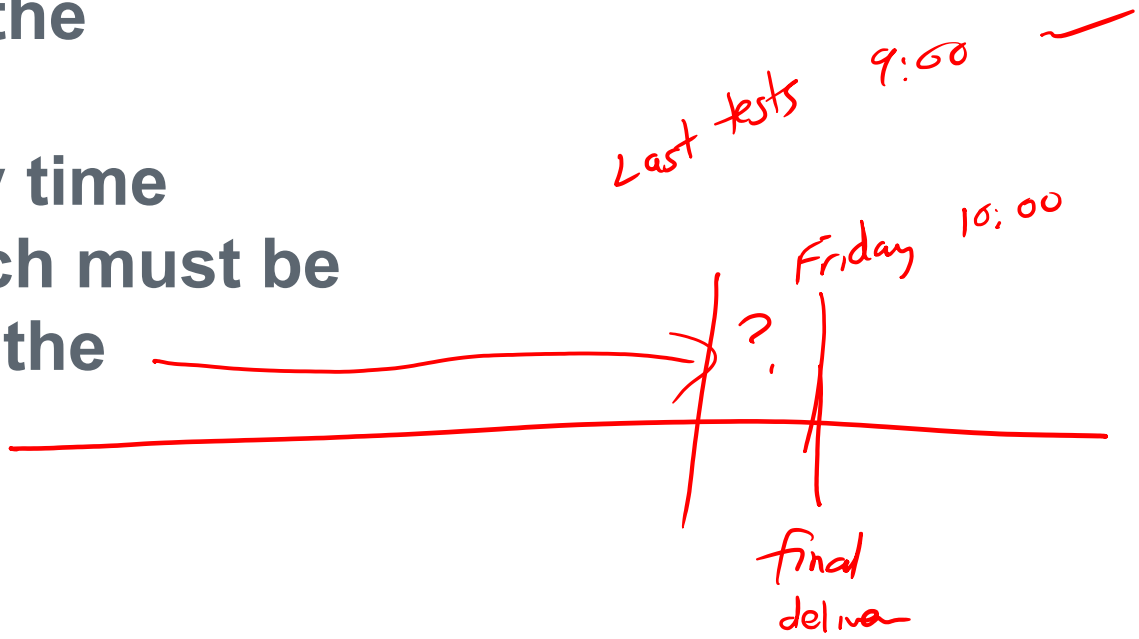
- Schedule can be documented in a Gantt chart
- Gantt chart identifies duration of tasks along with their starting and ending dates
- Gantt charts identify parallel tasks
- Multiple Gantt charts can be developed to show various levels of detail
  - Hierarchy of Gantt charts

Example Gantt Chart

Schedule Buffers

- Risk management must be used to guide the team in the amount of contingency time (buffer) which must be allocated to the schedule

- Schedule confidence is tied to the buffer



Summary