# Software Quality Assurance

## Module 9

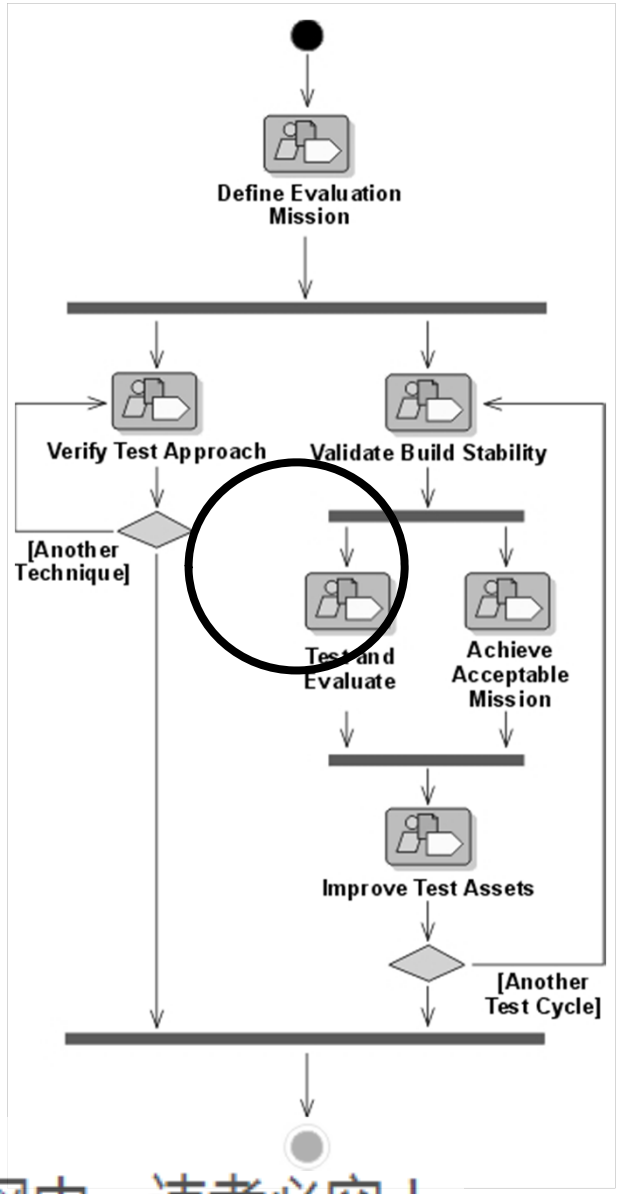
## Achieve Acceptable Mission

# Module 9 - Content Outline

- **Definition of the workflow:**
  ***Achieve Acceptable Mission***
- Reporting the status of testing

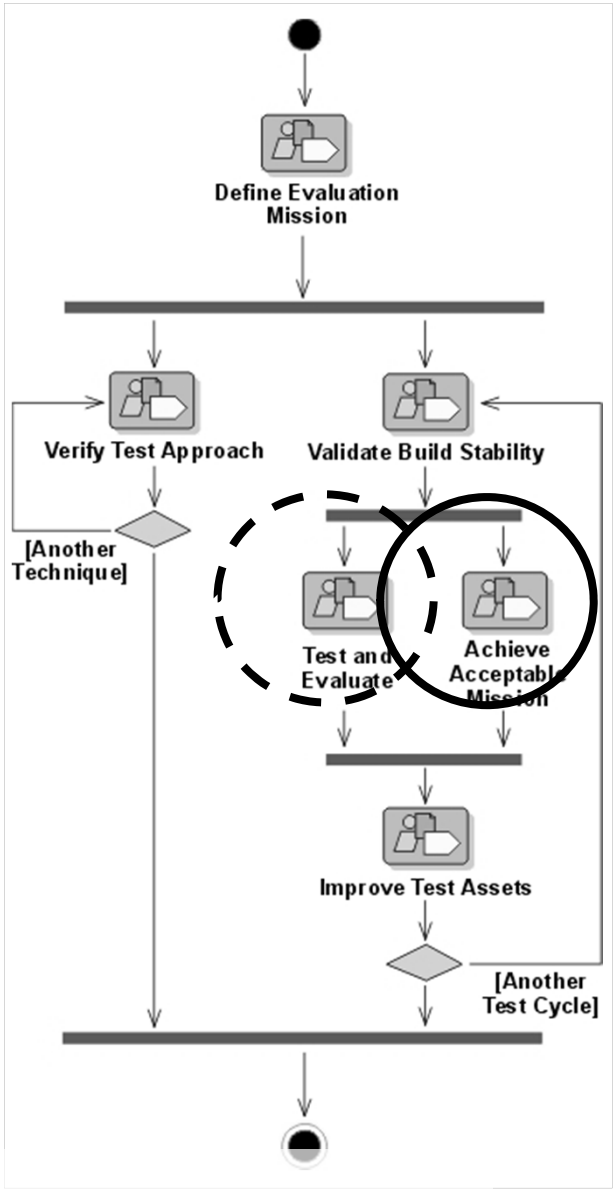
# Review: Where We've Been

- In the last two modules, we covered the workflow detail Test and Evaluate
- In part one, we looked at *techniques* for implementing tests, and in part two, we looked at *analyzing failures* and writing *change requests*.



# Achieve an Acceptable Mission

- In this module, we'll look at monitoring that the Mission is being achieved and reporting the status of the test effort



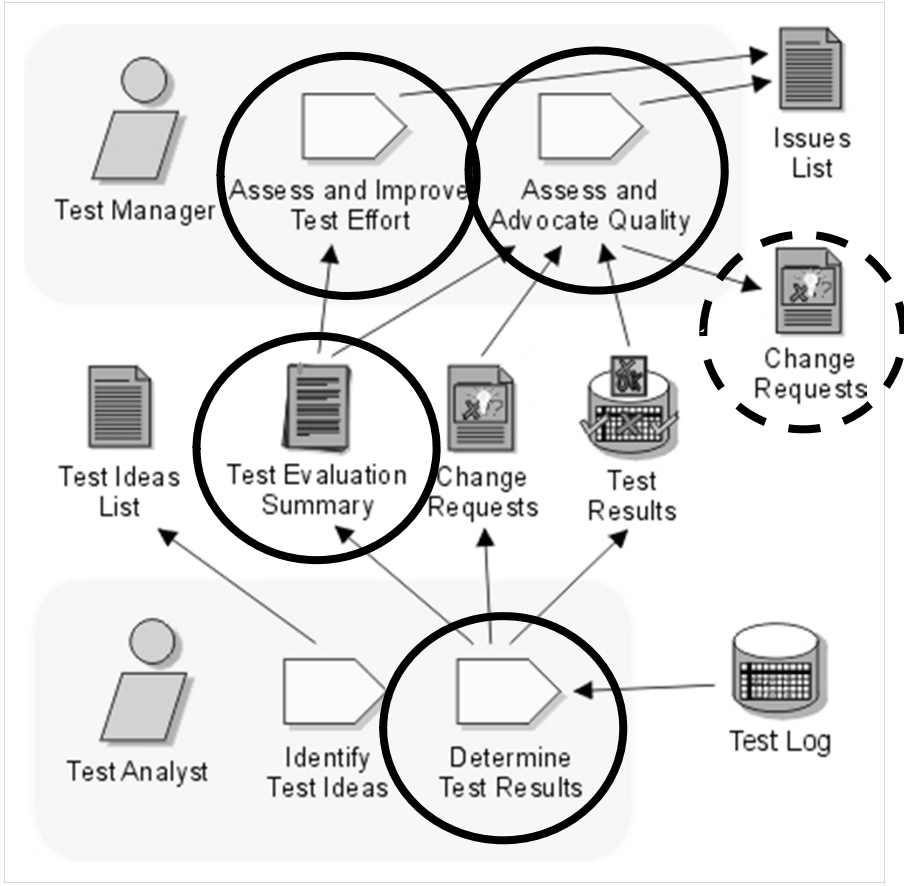
# Achieve an Acceptable Mission

- The purpose of this workflow detail :
  - To deliver a useful evaluation result to the stakeholders of the test effort—where useful evaluation result is assessed in terms of the Evaluation Mission. In most cases that will mean focusing your efforts on helping the project team achieve the Iteration Plan objectives that apply to the current test cycle.

- For each test cycle, this work is focused mainly on:
  - Actively prioritizing the minimal set of necessary tests that must be conducted to achieve the Evaluation Mission
  - Advocating the resolution of important issues that have a significant negative impact on the Evaluation Mission
  - Advocating appropriate quality
  - Identifying regressions in quality introduced between test cycles
  - Where appropriate, revising the Evaluation Mission in light of the evaluation findings so as to provide useful information to the project team

# Achieve an Acceptable Mission

- This module focuses on *Assessment* of the test effort reporting an *evaluation summary* of the test results
- In the last module, we discussed *change requests* which are used here to help evaluate status.
- We will look mainly at producing *Evaluation Summaries*.



# Module 9 - Content Outline

- Definition of the workflow:
  *Achieve Acceptable Mission*
- **Reporting the status of testing**

# Discussion Exercise 9.1: Reporting Status

- Pick a project and a point in time.
- You are the test manager.
- The project manager asks you:
  - How far are you with testing?
  - How much do you have left to do?
- How do you answer?

# Status Reporting

- Key questions: How far are we? How much is left to do?
- Experienced test managers have very different answers
- Complex, multidimensional question
  - Many types of data explain “extent of testing”
  - Simple metrics are often profoundly misleading
  - The best status reports consider several dimensions together
- Eight different categories of information

# Dimensions of “Extent of Testing”

Common answers are based on the:

- **Product** ◆ We’ve tested 80% of the lines of code.
- **Plan** ◆ We’ve run 80% of the test cases that we had planned to run.
- **Results** ◆ We’ve discovered 593 bugs.
- **Effort** ◆ We’ve worked 80 hours a week on this for 4 months. We’ve run 7,243 tests.

# Dimensions of “Extent of Testing”

- Common answers are based on the:

| | |
|---|---|
| **Obstacles** | We’ve been plugging away but we can’t be efficient until X, Y, and Z are dealt with. |
| **Risks** | We’re getting a lot of complaints from beta testers and we have 400 bugs open. The product *can’t be* ready to ship in three days. |
| **Quality of Testing** | Beta testers have found 30 bugs that we missed. Our regression tests seem ineffective. |
| **History across projects** | At this milestone on previous projects, we had fewer than 12.3712% of the bugs found still open. We should be at that percentage on this product too. |

# Status Reports – Extent of Testing

- Each dimension addresses a different issue
  - At times, each may be important to management
- Build status report around a cluster of dimensions
- Successful status reports provide range of different types of information, to give management a better context for decisions

# The Overall Structure of a Common Report

- Here's one structure that some managers find works well for them:
  - The report has four parts, each part starts a separate page.
  - Part 1  Risks and responsibilities
  - Part 2  Progress against plan or some other multidimensional chart
  - Part 3  Project bug metrics
  - Part 4  Deferred and no-fix bugs to approve

# The Overall Structure of a Common Report

- Part 1: Risks and responsibilities
  - Highlights current problems, such as:
    - Artifacts due into testing but not arrived
    - Artifacts that due out of testing but not yet completed
    - Staff turnover that threatens the schedule
    - Equipment acquisition problems that might threaten the schedule.
  - A project slips one day at a time
    - It can be recovered one day at a time
    - Encourage addressing the problems that cause slips
  - Good status reports show fine grain detail whenever it is likely that a reader could intervene and help the project, if only the reader understood (or was aware of) the problems that cry out for help

# The Overall Structure of a Common Report

- Part 2
  Progress against plan or multidimensional chart

| Component | Test Type | Tester | Total Tests Planned / Created | Tests Passed / Failed / Blocked | Time Budget | Time Spent | Projected effort for Next Build | Notes |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |

Elisabeth Hendrickson's report.

- Note how this covers progress against a plan, risks/obstacles, effort and results, all in one chart

# The Overall Structure of a Common Report

- Part 2
  Progress against plan or multidimensional chart

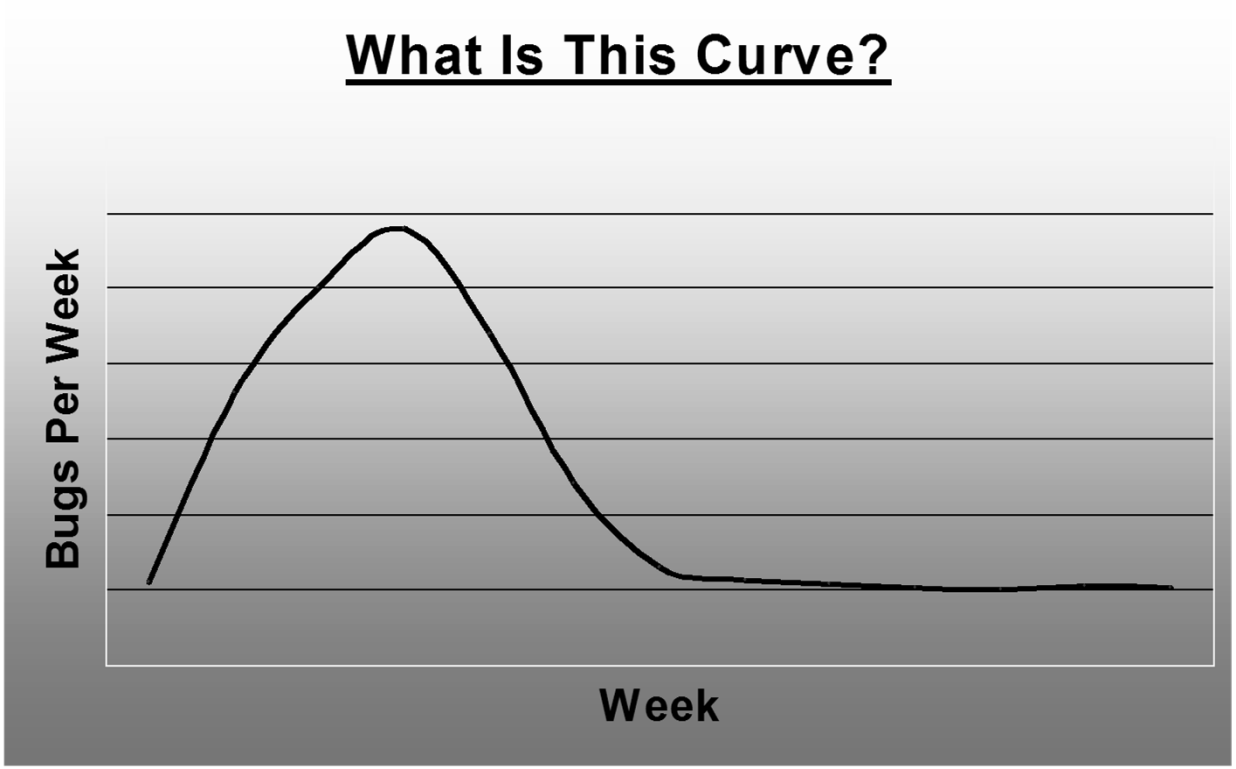
| Testing Dashboard | | | | Updated 11/1/00 | Build 32 |
|---|---|---|---|---|---|
| **Area** | **Effort** | **Coverage Planned** | **Coverage Achieved** | **Quality** | **Comments** |
| File/edit | High | High | Low | 😐 | 1345，1410 |
| View | Low | Med | Med | ☺ | |
| Insert | **Blocked** | Med | Low | ☹ | **1621** |

- James Bach's project dashboard
- Note how this covers areas, progress against plan, current effort, key results and risks, and obstacles.

# The Overall Structure of a Common Report

- Part 3
  Project bug metrics
  - These charts show find / fix rates over the course of the project.
  - Useful to give a sense of the rate at which problems are being repaired.
  - If the repair rate near the end of the project is slow compared to the find rate, the schedule is at risk.
  - It is too easy to over-interpret these charts

# Bug Counts and Extent of Testing?




- Some people believe they can measure testing progress by plotting a project's bug numbers against a theoretical curve of expected find rates over time.

# Potential Side Effects of Defect Curves

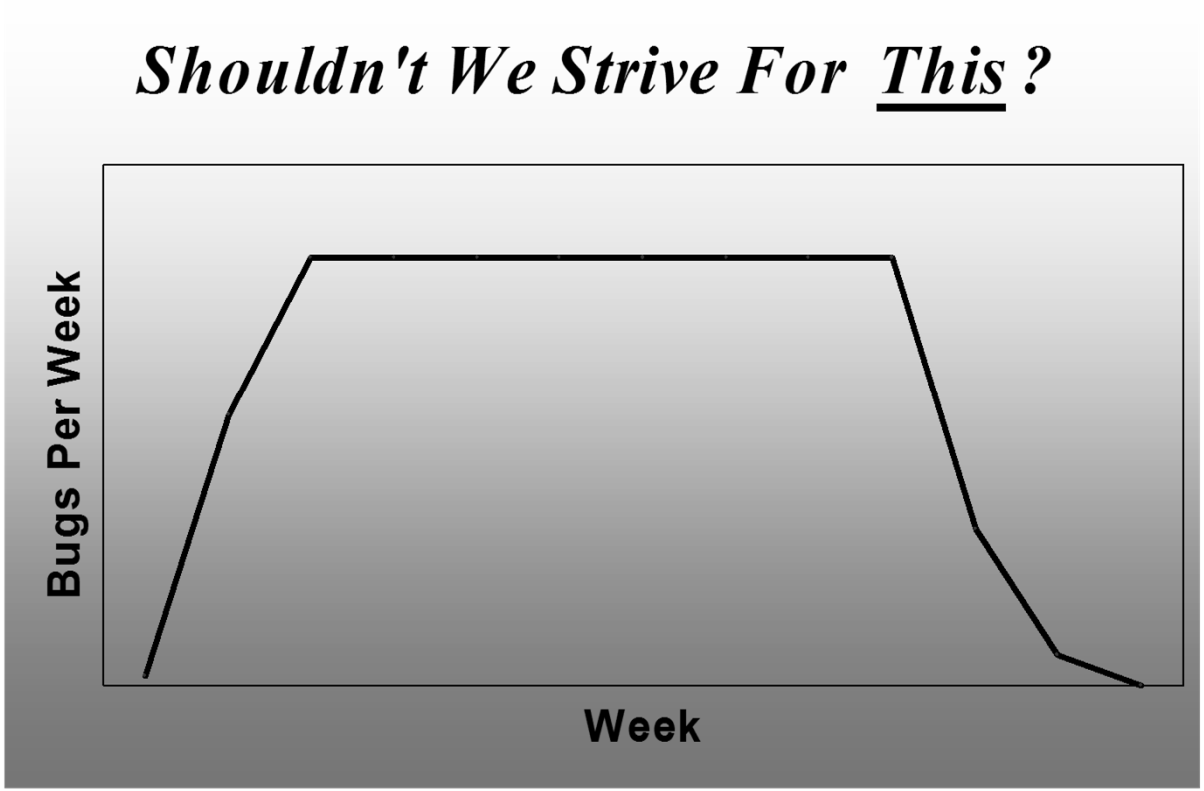
## Earlier in testing: Pressure is to increase bug counts

- Run tests of features known to be broken or incomplete.
- Run multiple related tests to find multiple related bugs.
- Look for easy bugs in high quantities rather than hard bugs.
- Less emphasis on infrastructure, automation architecture, tools and more emphasis of bug finding. (Short term payoff but long term inefficiency.)

# Potential Side Effects of Defect Curves

- Later in testing: Pressure to decrease find rate
  - Run lots of already-run regression tests
  - Don’t look as hard for new bugs.
  - Shift focus to appraisal, status reporting.
  - Classify unrelated bugs as duplicates
  - Class related bugs as duplicates (and closed), hiding key data about the symptoms / causes of the problem.
  - Postpone bug reporting until after the measurement checkpoint (milestone). (Some bugs are lost.)
  - Report bugs informally, outside of tracking system
  - Testers sent to movies before measurement milestones
  - Programmers ignore their bugs until reported by testers
  - Bugs are taken personally.
  - More bugs are rejected.

# Bug curve counterproductive?




- Sometimes, a drop in bug find rate reflects the declining efficiency of a given style of testing or overreliance on a specific technique.
- Perhaps the better solution, as bug rates drop, is to switch to a more powerful technique—such as one that had not yet been used because it relies on stability of individual features as a prerequisite event.

# The Overall Structure of a Common Report

- Part 4
  Deferred and no-change change requests
  - Every project team fixes some bugs and rejects or defers others.
    - At some point, there must be management review of the collection of problems that will not be fixed.
  - Rather than save up the list for the end of the project, list the new not-to-be-fixed change requests every week.

# Module 9 - Review

- Keep Status reporting frequent, simple and easy to understand.
- Select an appropriate way to measure the extent of testing.
- Use a standard reporting format that highlights important information appropriately.
- A “dashboard” is a useful summary tool.