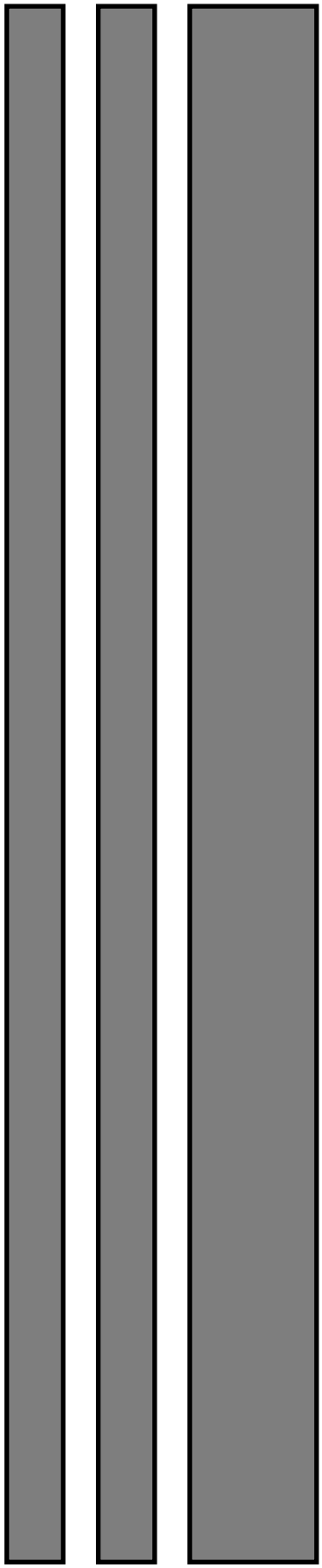


INSA

Introduction

Objectives

- Understand the basic concepts of life-cycle verification
- Learn basic terminology
- Understand the economic impact of early defect identification and repair
- Learn the difference between Quality Assurance and Quality Control

Synopsis

In this module, you will be introduced to the basic concepts of the review process. You will learn the economic reasons of why early defect detection and repair are so important to the success of a project. You will also learn how reviews fit into the software development process.

What are Reviews?

- Verification of interim deliverables
  - Requirements
  - Design documents
  - Models
  - Source code
  - Test plans
  - Use cases

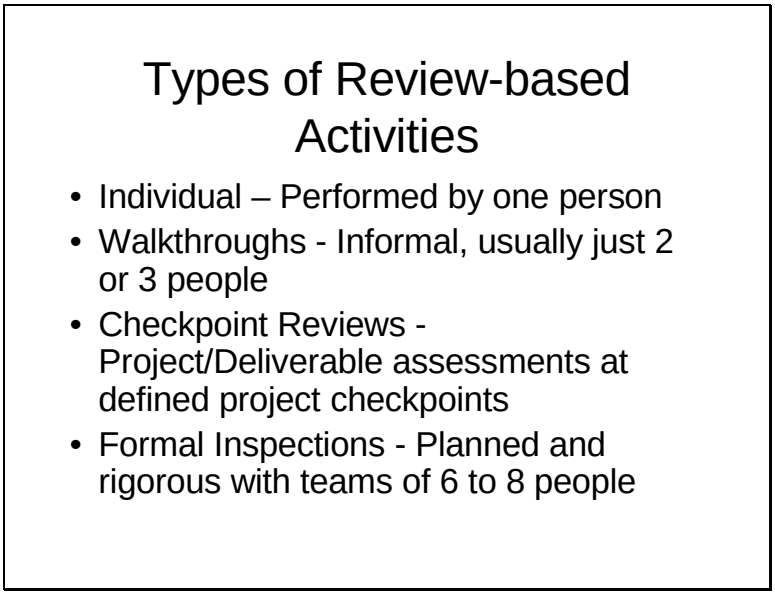
What are Reviews?

Reviews are very simply a verification of interim project deliverables, such as:

- Requirements
- Design documents
- Models
- Source code
- Test plans
- Use cases

Basically, anything produced on a project can and should be reviewed for correctness, usability, consistency and other quality factors.

In addition, other project characteristics can be reviewed to ensure certain project objectives have been met. Typically, this is the objective of the checkpoint review process.



Types of Review-based Activities

Reviews can be performed at differing levels of rigor.

- **Individual** – Performed by one person

  Individual reviews may take the form of desk checking by the author, or an individual review by a peer.

- **Walkthroughs** - Informal, usually just 2 or 3 people

- **Checkpoint Reviews** - Project/Deliverable assessments at defined project checkpoints

- **Formal Inspections or Reviews** - Planned and rigorous with teams of 3 to 7 people

Individual Reviews Compared to Team Reviews

- Individual Reviews are Helpful When:
  - The risk is low
  - Access to subject matter expertise and other reviewers is limited
  - People actually will take the time to review the deliverable
  - People are good at finding problems
- Team Reviews are Helpful When:
  - The risk is moderate to high
  - Multiple perspectives are needed
  - You want to find the most number of issues
  - You have a culture of trust and communication

Individual Reviews Compared to Team Reviews

There are pros and cons to each type of review. Generally, a team review will identify more issues than a solo effort.

Individual Reviews are Helpful When:

- **The risk is low**

  If errors do not have a big impact, then individual reviews may be adequate.

- **Access to subject matter expertise and other reviewers is limited**

  If there are only a few people that are qualified or knowledgeable to review something, they may not have the time to attend a lot of meetings.

- **People actually will take the time to review the deliverable**

  You are at the mercy of people actually taking the time to carefully examine the product. Some people may wait until the last minute and then just give a quick look at something, which is a bad thing.

- **People are good at finding problems**

  Some people have a better eye for detail and can think of more questions than others.

Team Reviews are Helpful When:

- **The risk is moderate to high**

  For example, in the case of a critical requirements document, an error can impact design, coding, testing and many other things.

- **Multiple perspectives are needed**

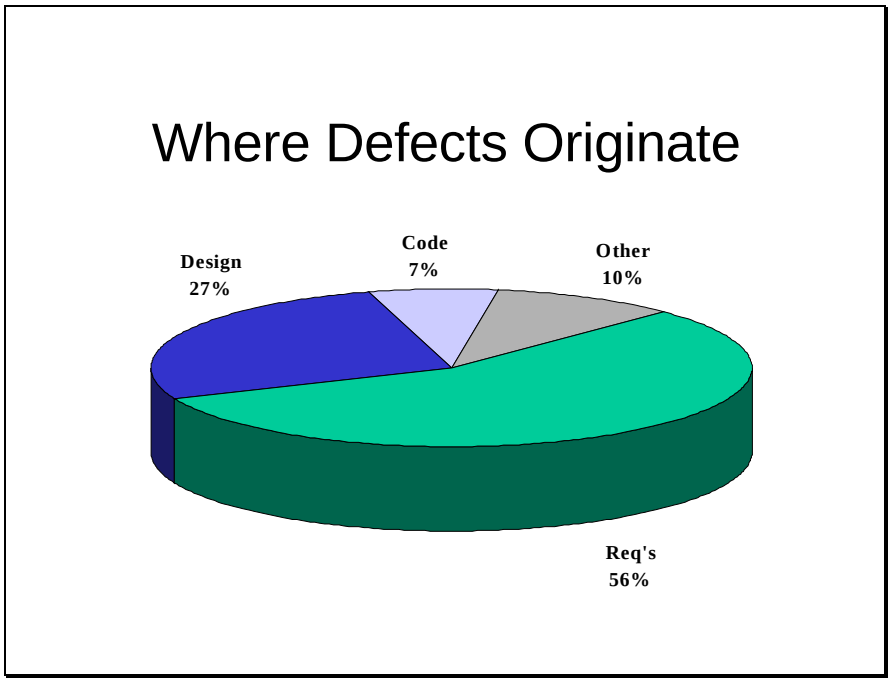
  Sometimes you need the input from several people to get a good review. For example, a test plan may need to be examined by testers, users and developers.

- **You want to find the most number of issues**

  Since more people are involved, a team review will often find more issues than even several individual reviews. Part of the reason for this is that in discussing the item, people may think of more issues.

- **You have a culture of trust and communication**

  If people are not fearful of speaking or of having others discuss their work, team reviews can be good. However, if people are fearful, then individual reviews are less intimidating. Keep in mind, that feedback can be destructive in both methods if not handled well.



Why Perform Early Verification?

There is a definite economic impact of early verification, such as reviews. One economic impact is from the cost of defects. This is a very real and very tangible cost.

Another economic impact is from the way we find defects. It is possible to have very good motivations and goals while achieving them in a very inefficient way.

In this section, we will examine the economic impact of defects and ways to economize the overall software development process.

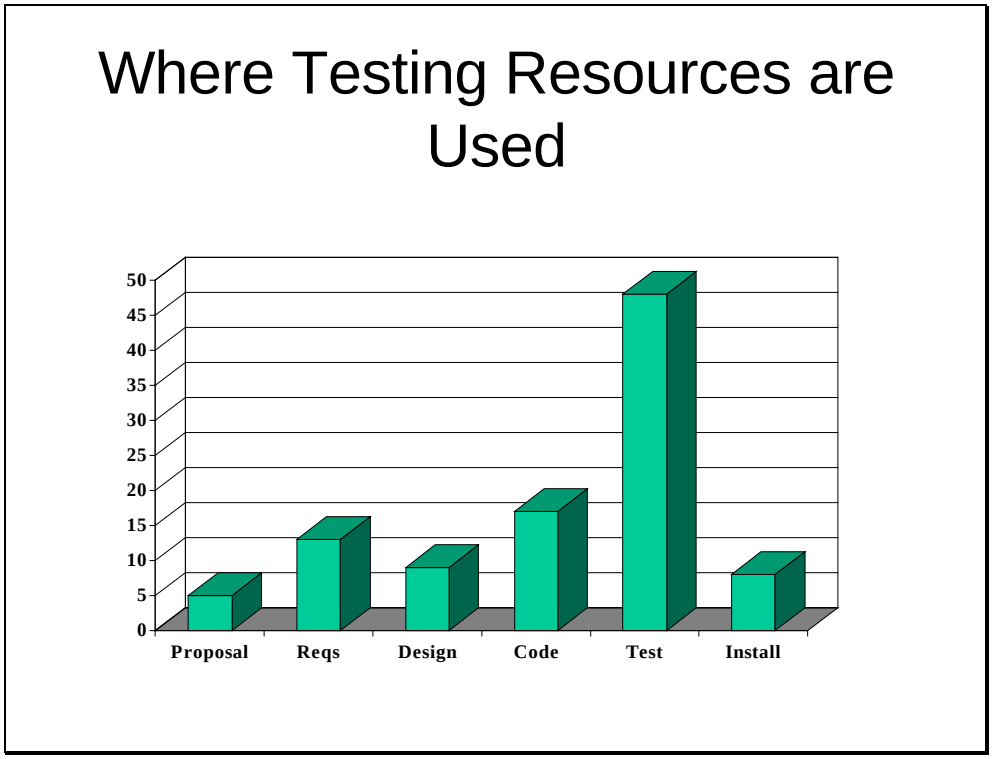
Where Defects Originate

To understand the dynamics and costs of defects, we need to know some things about them. One of the most commonly understood facts about defects is that most defects originate in the requirements definition phase of a project. The next runner-up is the design phase.

Some problems in getting accurate, clear, and testable requirements are:

- Many people do not have a solid requirements gathering process
- Few people have been trained in or understand the dynamics of requirements
- Projects, people, and the world around us change very quickly
- The English language is ambiguous and even what we consider clear language can be interpreted differently by different people.

The figures in this pie chart were taken from a James Martin study and the numbers track very closely to measurements of typical software projects.

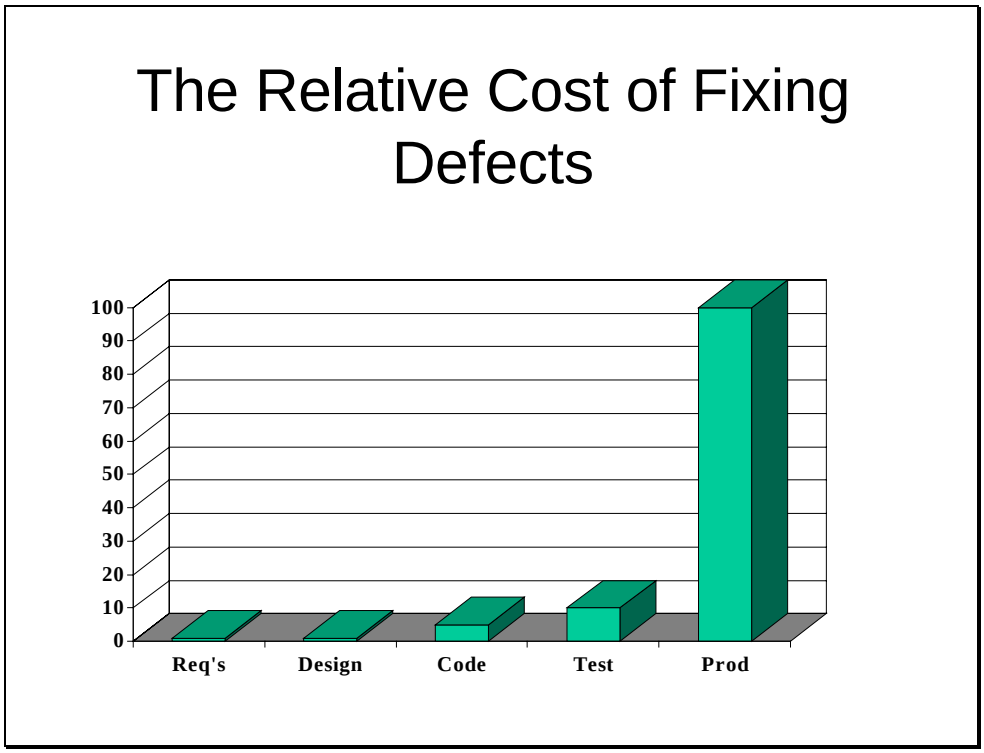


Where Testing Resources are Used

We saw that most defects originate in requirements and design, but most of the testing effort occurs in a traditional "testing" phase toward the end of the project. This is called the "big bang" approach from the concentration of effort at one big phase. "Big bang" could also describe the sound of the project as it fails.

The problem with the big bang approach to testing is that defects are not found until toward the end of the project. This is the most costly and risky time to fix defects. Some complex defects may even be impossible to fix.

The figures above were taken from surveys at the Quality Assurance Institute's (QAI) annual software testing conference.



The Relative Cost of Fixing Defects

One of the most well known facts about software defects is that the longer they go undetected, the more expensive they are to fix. Although research differs on the exact ratios, the general rule is 1:10:100.

That is, if a defect costs one unit (hour, dollar, etc.) to fix in requirements and design, it costs 10 units to fix in testing (system/acceptance) and over 100 times to fix in production. Sometimes the cost to fix a defect in production costs much more than 100 times the cost of fixing it in the requirements phase.

This cost of defects doesn't even take into account the impact cost of defects. These costs could be attributed to lost revenue, reimbursements, fraud, lost customers, bad public relations, and litigation. In the case of safety critical systems, how can one put a cost value on a human life?

The Bottom Line

- Most defects are created in the early stages of a project
- Most defects are found in the later stages of a project
- It costs 10 to 100 times as much to fix a defect in the later phases of a project.

The Bottom Line

So, what does all of this mean? The main conclusion is that most people perform testing too late in the process. These people wonder why testing is so expensive and why their projects are often over budget.

If you really want to make your testing more efficient and reduce the overall cost of testing and defects, test early in the project and continue testing throughout the project. Most defects can be found by inspections before a test is even performed!

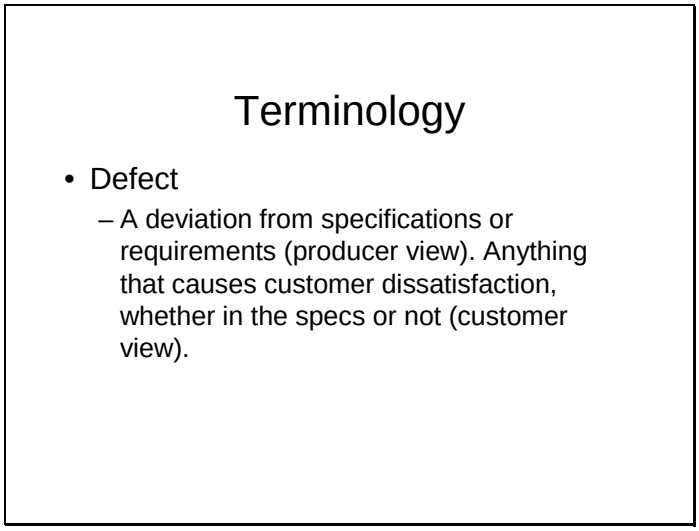
The Bottom Line

- If you want to reduce the cost of testing, spend time early in the system development (or purchase) process to make sure the requirements and design are correct.

The Expectation Gap

Managing Expectations

The above picture shows what is known at the expectation gap. This gap is the difference between what someone expects and what they actually get. The wider the gap, the greater the disappointment. So, obviously, we should strive to keep the gap small. This is done by keeping the customers and users involved in the project so there will be no last-minute surprises. This is not a test task as much as it is a project management task.

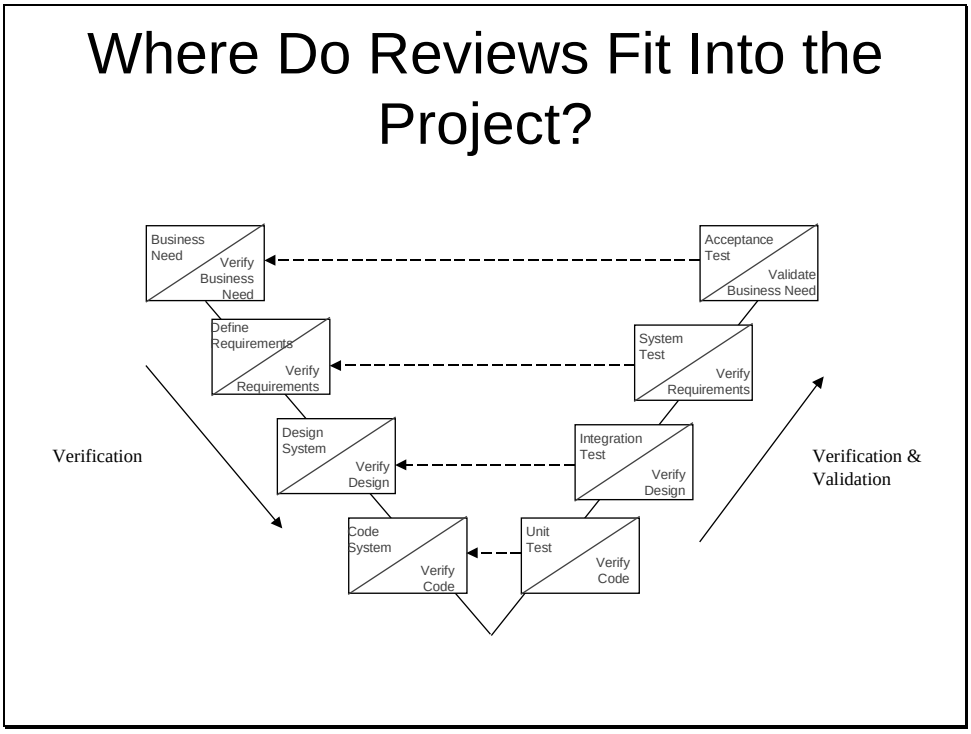


Test Terminology (Cont'd.)

- **Defect**

A deviation from specifications or requirements (producer view). Anything that causes customer dissatisfaction, whether in the specs or not (customer view).

In some organizations, defects are called problems, incidents, anomalies, trouble reports, etc. Whatever the terminology, a defect is a defect.



Test Terminology (Cont.)

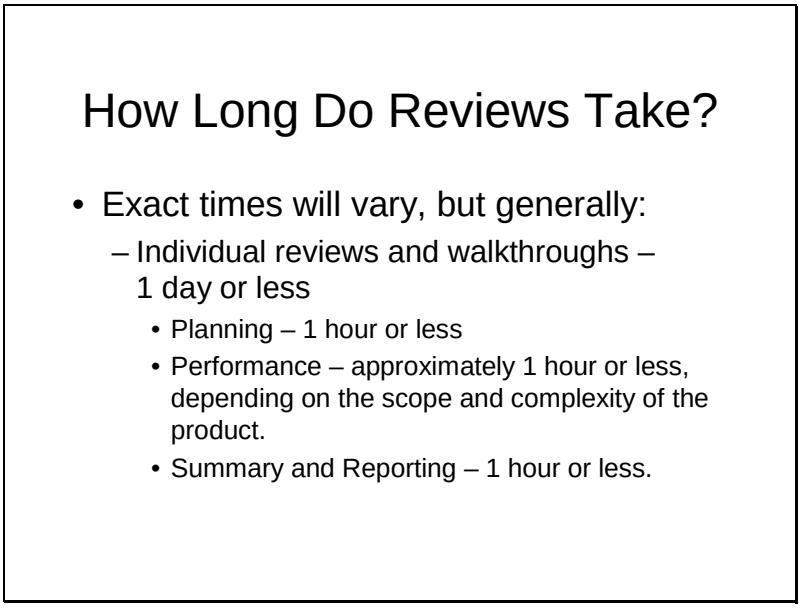
Where Do Reviews Fit into the Project?

In this diagram, the major phases of development are shown along with the corresponding phases of testing. The order of execution is also indicated from the upper left to the bottom of the "V" and back up to the upper right.

Each box has either a verification step (a test performed by inspecting something) or a validation step (a test performed by executing software on the computer).

The "V" diagram can be used to depict testing in any methodology, including Rapid Application Development.

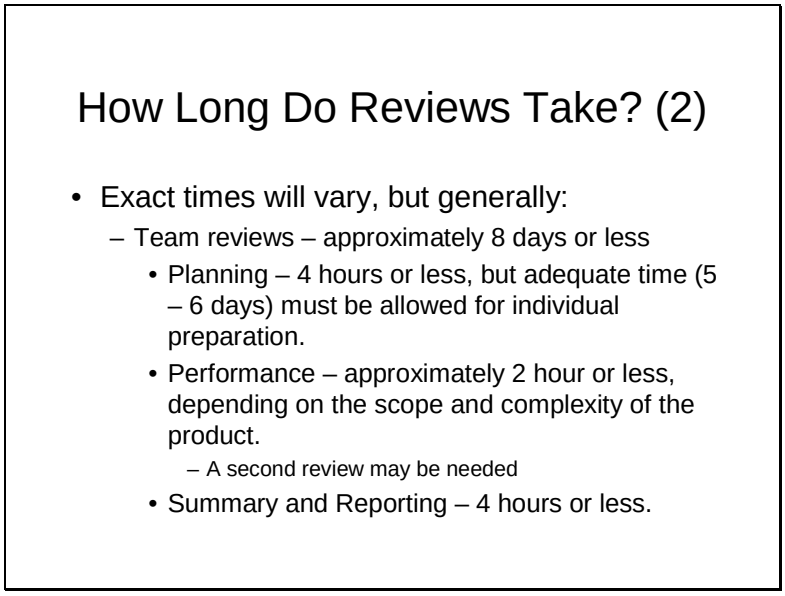
It is important to note that user acceptance testing is at the top of the "V" and validates business or operational need, not that the system was built according to requirements.



How Long Do Reviews Take?

One of the major objections to performing reviews is that they "take too long". Research has shown, however, that reviews save more time on a project than they consume in performing them.

Reviews can take as little or as much time as you want to devote to them. For the purpose of guidelines, individual reviews take the least amount of time because it is just one person looking at something. There is very little to plan and coordinate between people. So, for the most part you can do a solo review in three hours or less.



How Long Do Reviews Take? (2)

For team reviews, exact times will vary, but generally the process takes about 8 days or less:

- **Planning** – 4 hours or less, but adequate time (5 – 6 days) must be allowed for individual preparation.
- **Performance** – approximately 2 hour or less, depending on the scope and complexity of the product. Keep in mind that a second review may be needed
- **Summary and Reporting** – 4 hours or less.

Interpersonal and Cultural Issues

People are often reluctant to embrace reviews and inspections because:

- **They think there is not enough time**

Actually, reviews increase the chances of meeting the project deadline. This finding has been shown repeatedly from major organizations that have published their experiences in implementing reviews.

- **The information will be used against them**

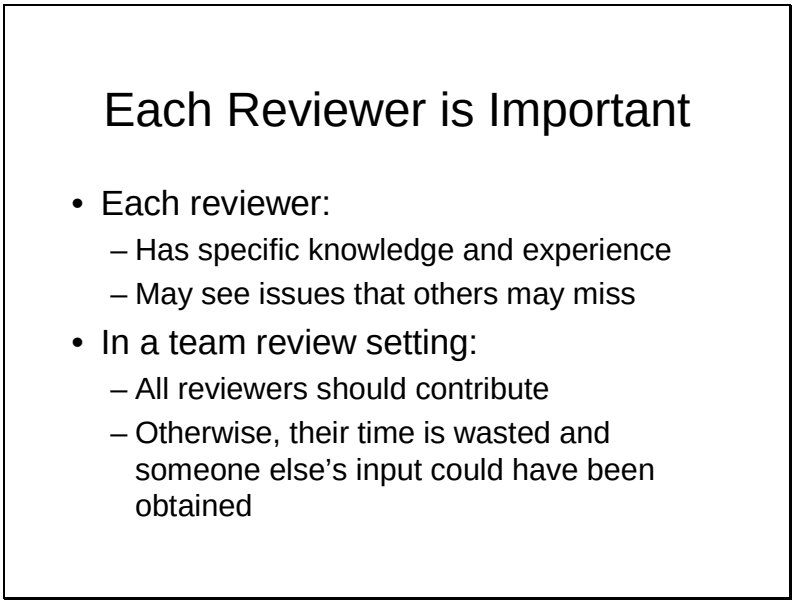
The best way to kill a review effort is to use the information for punishing people. People will likely be skeptical at first. That is why a culture of trust needs to exist for people to totally embrace reviews.

- **They are afraid of personal criticism**

This is a natural fear. Few people enjoy having their work openly critiqued. Training for reviews must emphasize that the focus of the review is the product and not the producer. However, this takes practice.

- **Pride of creative ownership**

People naturally have a strong attachment to the work they create. This is why ego often plays a major part in how people approach the review process. Ideally, egos should be checked at the door. Realistically, egos can get bruised by careless remarks about either the product or the producer.



Each Reviewer is Important

Reviewers should not take their role lightly. Reviewers are often the only chance to find defects before the next stage of the project.

Each reviewer:

- **Has specific knowledge and experience**

  This is the reason a reviewer is invited to a review session. The only exception is when reviews are used for training or exposure to new information.

- **May see issues that others may miss**

  The more people that review something, the more changes of finding something.

In a team review setting:

- **All reviewers should contribute**

  Otherwise, their time is wasted and someone else's input could have been obtained.

What to Look For

- Incorrectness
- Vague, confusing or ambiguous descriptions
- Inconsistencies
- Incompleteness

What to Look For

In this course you will find several checklists, each for different types of deliverable products, such as requirements, design, code, test plans, etc.

Generally speaking, you are looking for:

- **Incorrectness**

  These are errors that will impact the correctness of something. For example, incorrect formulas, rules, and calculations.

- **Vague, confusing or ambiguous descriptions**

  These are items that are unclear or confusing. These can also lead to assumptions and mistakes.

- **Inconsistencies**

  Sometimes items will be in conflict. For example, one requirement may specify a value that is different than specified in another requirement.

- **Incompleteness**

  Gaps and omissions are common occurrences in deliverable. For example, a requirements document may specify what to do in a situation, but may fail to specify what to do if the situation does not occur.

What Can Be Deferred

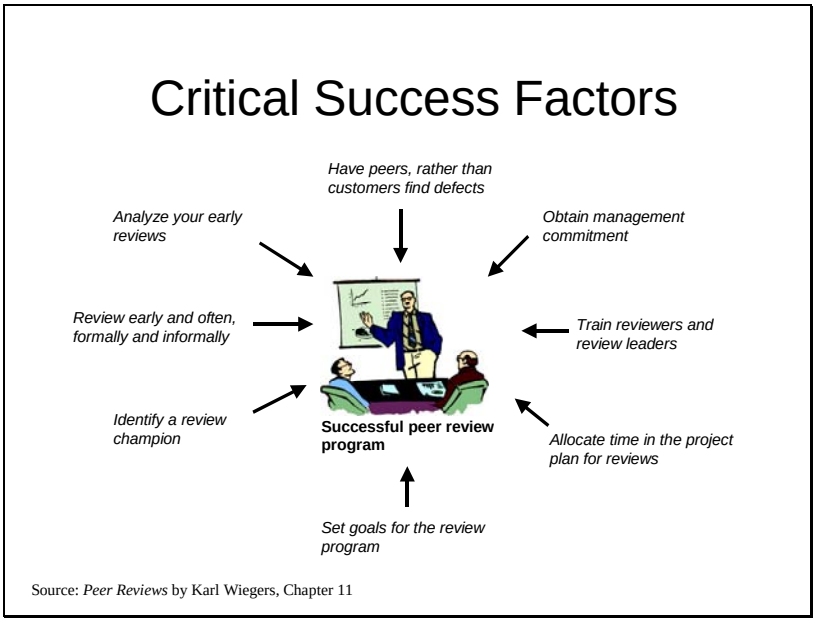
- Typos
- Style differences
- Formatting
- As long as these do not cause incorrectness or other negative impacts

What Can Be Deferred

Minor issues such as typos and formatting can be addressed in a side list as not to distract from the more important and substantial issues. At the end of this module is a template for recording typos.

The main thing to consider is whether or not these issues cause incorrectness or other negative impacts. For example, a file name may be spelled incorrectly in a design document, which could have a major impact in later project phases. This type of error would need to be considered a defect. On the other hand, a typo in a requirements document, such as “whan” instead “when” is obvious and does not take away from the main meaning.

In Chapter 11 of "Peer Reviews in Software" you will find additional guidance in achieving these success factors.



Critical Success Factors

- **Have peers, rather than customers, find defects.**

Customers can be anyone who sees or depends on the work you produce.

- **Obtain management commitment**

Management needs to be in the driver's seat and must be out in front in making the message that reviews are important.

- **Train reviewers and review leaders**

Training is a good foundation, but needs to be reinforced with experience.

- **Allocate time in the project plan for reviews**

If people do not have time built into the schedule for review, reviews will naturally fall by the wayside.

- **Set goals for the review program**

Ideally, the goals will be achievable and measurable.

- **Identify a review chairperson**

The process must have an owner that is visible, capable and responsible.

- **Review early and often, formally and informally**

Even quick and informal reviews are better than nothing.

- **Analyze your early reviews**

Take advantage of lessons learned early.

Software Inspection Best Practices

- Plan inspections to address your project and inspection objectives.
- Use serious, quantitative entry and exit conditions.
- Inspect upstream documents first.
- Begin inspecting documents early in their lives.
- Check against source and related documents.

Source: Gilb 1998, 2000

Software Inspection Best Practices

- **Plan inspections to address your project and inspection objectives.**

  This gives a good definition of the scope of reviews.

- **Use serious, quantitative entry and exit conditions.**

  This prevents defects from creeping from one project phase to the next.

- **Inspect upstream documents first.**

  This allows defects to be caught early.

- **Begin inspecting documents early in their lives.**

  It's OK to review something before it is complete.

- **Check against source and related documents.**

  It's good to cross-reference documents for consistency and correctness.

Software Inspection Best Practices (cont'd.)

- Prepare and inspect at your organization's optimum rates.
- Focus on major defects.
- Measure your benefits from inspections.
- Emphasize defect prevention and process improvement.

Software Inspection Best Practices (cont'd.)

- **Prepare and inspect at your organization's optimum rates.**

  Don't slow down your current processes.

- **Focus on major defects.**

  Follow the 80/20 rule, especially at first.

- **Measure your benefits from inspections.**

  This will justify this and future endevors.

- **Emphasize defect prevention and process improvement.**

  The goal is improvement, not criticism.

Review Traps to Avoid

- Participants don't understand the review process.
- The review process isn't followed
- The right people do not participate
- Review meetings drift into problem-solving
- Reviewers focus on style, not substance
- Reviews are seen as a formality
- Management abuses the information from reviews
- Reviews take on a critical tone

Adapted from *Peer Reviews* by Karl Wiegers, Chapter 11

Review Traps to Avoid

- **Participants don't understand the review process.**

  Solution – Training and experience

- **The review process isn't followed**

  Solution – QA group to determine compliance to the process

- **The right people do not participate**

  Solution – Show people how lower defects makes a positive difference to them

- **Review meetings drift into problem-solving**

  Solution – Moderator keeps the review meeting on track

- **Reviewers focus on style, not substance**

  Solution – Moderator keeps the reviewers focused on finding defects

- **Reviews are seen as a formality**

  Solution – People understand why reviews are performed

- **Management abuses the information from reviews**

  Solution – Management understands that abuse of information will kill the reviews effort.

- **Reviews take on a critical tone**

  Solution – Moderator keeps the session on a positive tone.