

**Instructor Notes:**

Defect Reporting and Defect  
Life Cycle Management

Lesson1: Defect Free Defect Reporting

**Instructor Notes:****Lesson Objectives**

To understand the following topics:

- What is Software Quality?
- Defect Definition
- Why find Defects?
- Impact of Defects
- Legal Implications
- Life-cycle workflow
- Life-cycle workflow - Enhancement
- Defect Report - Definition
- Defect Reporting – The Need
- Defect Report - Template
- Important Attributes
- Example on Severity & Priority



2

**Instructor Notes:****Lesson Objectives**

To understand the following topics:

- Defective Reports - Certain Facts
- Importance of Effective Defect Reporting
- Defect Free Report - Recommendations
- Writing Defect Free Reports
- Preparation
- Reporting and Communication Process
- Guidelines
- Using Tools for Reporting - Advantages
- Project/Organization Level Process
- Project/Organization Level Process
- Defect Free Reports - Advantages
- Nine Commitments worth making to developers

**Instructor Notes:**

### 1.1 Software Quality

#### What is Software Quality?

Quality of the developed software exhibits the following aspects :

- It is reasonably bug or defect free
- Delivered on time and within budget
- Meets requirements and is maintainable

ISO 8402-1986 standard defines quality as "Totality of features or characteristics of a product or service that bear on its ability to satisfy Stated and Implied needs

Acceptance criteria defines what minimum requirements should be met

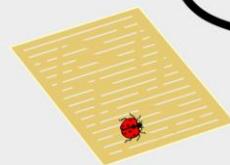


**Instructor Notes:****1.2: Defect Definition**

A Software Engineer makes an error...



...that creates a defect in the software...



...that can cause a failure in operation



**Instructor Notes:****1.2: Defect Definition****Defect**

- If AUT's (Application Under Test's) some feature or function is not working as per what is there in requirement it is called as defect
- A defect is a variance from a desired product attribute.
- A problem which, if not corrected, could cause an application to either fail or to produce incorrect results

**Examples**

- Car brakes stop working after crossing speed of 100 km/hr
- Billing software generates, prints, and mails bills showing amount payable as 0 Rupees
- A customer goes to withdraw \$1000 from his account having a balance of \$5000 and minimum bank balance require is \$500. ATM Rejects the request saying "Insufficient Balance"
- Actual amount withdrawn from the ATM and the amount printed on print receipt shows difference

**Instructor Notes:****1.2: Defect  
Why find Defects?**

- Increase confidence in the reliable operation of the system and get more business
- Reduce the likelihood of loss or even life-threatening incidents
- Obtain repeat and referral business from satisfied customer
- Decrease overall system costs associated with quality problems

**Instructor Notes:****1.2: Defect Impact of Defects**

A single error can cause nothing or a lot

It can cause death or injury if it fails in case of safety critical applications

It can also cause huge financial loss to clients

And also lead to fine/penalties for us

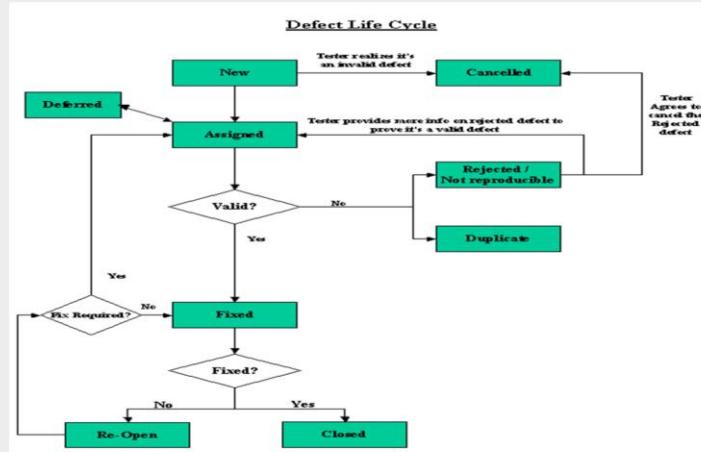
**Examples**

- NYSE fined Waterhouse Investor Services US \$225,000 for its web site failures - inability to file on-line stock orders and inadequate customer service
- An AA jet crashed in Colombia because the captain entered an incorrect one-letter computer command that sent the jet into a mountain killing 158 people aboard. When there is critical command, the software could have asked for confirmation or verified or have enough validation before processing the command.
- Hacker hacked into US government computers, including two agencies within the Defense Department, and defaced government Web sites. It shows the insufficient Security Testing.

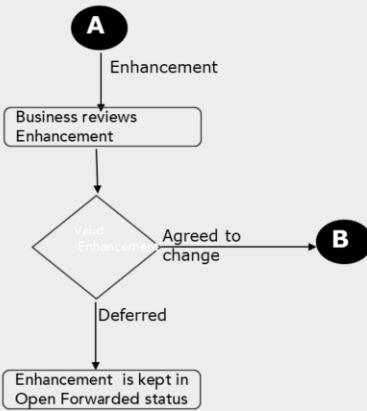
**Instructor Notes:****1.2: Defect Legal Implications**

- Under a contract, a buyer can sue a company if he/she did not get what he/she paid for or if the software did things it was not supposed to do
- Buyer can also sue for Consequential damages - economic loss or injury to person or property
- Acts like Data Protection Act also safeguard the security rights of the customer



**Instructor Notes:****1.2 Defect Life-cycle workflow**

1. **New** – When a Defect is logged and yet to be assigned to a developer. Usually Project Manager or Dev Lead will decide on which defects to be assigned to which developer.
2. **Assigned** – indicates that the developer who would fix the defect has been identified and has started analyzing and working on the defect fix.
3. **Duplicate** – Manager or Developer will update the status of a defect as “Duplicate” if this defect was already reported.
4. **Rejected / Not Reproducible** – This status indicates that the developer is not considering the defect as valid due to following reasons
  - a) Not able to reproduce
  - b) Not a valid defect and it is as per requirement
  - c) Test Data used was invalid
  - d) Defect referring to the Requirement has been de-scoped from the current release, tester was not aware of this late changes.
5. **Deferred** – Defect fix has been held back because of time or budget constraints and project team has got approval from customer to defer the defect till next or future release.
6. **Fixed** – Developer has fixed the defect and has unit tested the fix. The code changes are deployed in test environment for verifying the defect fix.

**Instructor Notes:****1.2 Defect  
Life-cycle workflow - Enhancement**

**Instructor Notes:****1.3: Defect Report Definition****Defect report**

- Is a document to maintain all the defects, that test engineer found while test execution
- The most important deliverables to come out of test. It will have more impact on the quality of the product than most other deliverables from test
- It is important to write effective defect reports

**Instructor Notes:****1.3: Defect Reports**  
**Defect Reporting – The Need**

- Emphasize on continuous improvement
- Defect report – an important deliverable
- Inadequate Material
- High impact of defective defect report

**Instructor Notes:**

### 1.4 Defect Report Template

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Project ID:	Project Name :												
Defect Id.	Module name	Defect Summary	Defect Description	Defect Category	Detected in Browser	Environment	Defect Severity	Defect Priority	Detected in Release #	Detected in Build #	Reported Date	Reported By	Assigned To
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													

[Project Profile](#) | 
 [Defect Tracking Sheet](#) | 
 [Revision History](#) | 
 [Help](#)

- **Defect ID** - A unique number to each defect. This will help to identify the bug record. If you are using any automated bug-reporting tool then this unique number will be generated automatically each time you report the bug.
- **Module Name**- It will contain the name of the module being tested.
- **Defect Summary**- A short summary of the defect. It can be in 1 or 2 lines.
- **Defect Description**- A detailed description of bug. It includes
  1. Clearly mention the steps to reproduce the bug.
  2. How application should behave on above mentioned steps.
  3. What is the actual result on running above steps i.e. the bug behavior.
- **Defect Category**- Categorizing the defect into the appropriate category. The categories considered are:
  - Coding: When the defect is found in the code.
  - Design :When the defect is found in the design
  - Enhancement: If the defect stated is actually an enhancement to the present requirement
  - New Requirement :If the defect stated is actually a new requirement
  - Query : is any question or doubt which might be raised by the tester, it need not be an actual defect.
  - Documentation: is any error found in the documents of the application like the help document etc.
  - Master Data: Is any error found in the master data received
  - Test Review: is while doing the testing any review comments which might be suggested by the tester ,again it may not be a defect itself.
  - DO – Documentation related defects
  - This includes all defects related to missing or misstated requirement in Functional specification
  - BD/PK/LD - Build / Package / Load
  - Change management, library, version control.
  - FN/LO (Coding)
  - Function/ Program Logic (Logic, pointers, loops, recursion, computation); Not functioning as per the design / requirement.
  - EN - Environment
  - Environment (Design, compile, test, or other support system problems)
  - OP
  - If the optimal usage is lacking or performance is not up to the mark. Design will adversely affect the product's performance
  - SG – Suggestion
  - If the defect reported is a suggestion.

**Instructor Notes:****1.4 Defect Report Template(contd.)**

Detected in Release #	Detected in Build #	Reported Date	Reported By	Assigned To	Status	Review type / Test Cycle	Test Case No.	Fixed By	Fixed Date	Verified Date	Verified By	Verified in Release #	Verified in Build #	Attachments	Comments
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															

Project Profile | Defect Tracking Sheet | Revision History | 

- **Detected in Browser :** The name of the browser the defect was found in (IE, Firefox ,etc.)
- **Environment:** Mention the environment where the defect was found. It can be internal Dev,internal QA,External Dev or External QA.
- **Defect Severity :** Impact of the defect on functionality of application. The values it can contain :
  - P0 defect is a defect wherein the main function of the software does not work. e.g. crash, hang, data corruption
  - P1 defect is a defect wherein a function does not work and a tedious work around exists. e.g. One of menu options does not work.
  - P2 defect is a defect wherein the software does useful work but a degree of inconvenience is caused. Correction is not deferrable and easy work around exists.
  - P3 defect is a tolerable defect as corrections are deferrable. E.g. cosmetic problems in user interface like spelling.
  - “High” severity defect is a defect wherein the main function of the software does not work. e.g. crash, hang, data corruption, some of the menu functions do not work
  - “Medium” severity defect is a defect wherein the software does useful work but a degree of inconvenience is caused. Correction is not deferrable and easy work around exists.
  - “Low” severity defect is a tolerable defect as corrections are deferrable. E.g. cosmetic problems in user interface like spelling.

**Instructor Notes:**

### 1.4: Defect Reports

#### Important Attributes

- Defect ID
- Module name
- Defect summary
- Defect description
- Defect Category
- Detected in Browser
- Environment
- Defect Severity
- Defect Priority
- Detected in Release #
- Detected in Build #
- Reported Date
- Reported By
- Assigned To
- Status
- Review type / Test Cycle
- Test Case No.
- Fixed By
- Fixed Date
- Verified Date
- Verified By
- Verified in Release #
- Verified in Build #
- Attachments
- Comments

- **Priority** – It indicates When bug should be fixed? The values are low, medium and high
  - High - This bug should be resolved as soon as possible in the normal course of development activity, before the software is released.
  - Medium - This bug should be repaired after serious Defects have been fixed.
  - Low - It can be resolved in a future major system revision or not be resolved at all.
- **Detected in Release #** : Denotes the Release # in which the defect was detected.
- **Detected in Build #**: Denotes the Build # in which the defect was detected.
- **Reported Date**: The date on which the defect was reported. It need not be same as the date on which it was detected.
- **Reported By**: Name of the person who reported the defect.
- **Assigned To**: Name of the person the defect is assigned to , to fix it.
- **Status**: Gives the status of the defect. It can contain the following values
  - Verified
  - Closed
  - Fixed
  - Active
  - CNR-Could Not be Reproduced
  - NOD: reported bug Not a Defect
  - REP: Repeated bug
- **Review type / Test Cycle**: It can contain following values:
  - PPR-Peer to Peer Review
  - PR-Peer Review
  - SR-Self/Programmer Review
  - QPR-10% Quality Probe(Review)
  - RT-Random Testing
  - UTC-Using Test Cases

**Instructor Notes:****1.5: Defect Reports**  
**Example on Severity & Priority**

Think of the following type of problem:

A spelling error on a user-interface screen

- What severity and priority does this issue deserve?

Well, judging from our earlier definitions, it would seem that this is a low-severity item. After all, the server doesn't crash due to a spelling error.

But is this truly a low-severity problem?

A spelling error will probably not hinder a customer's ability to use the system, but it greatly affects the customer's perception of the company that created the product and of the quality of the product. So from customer-relations and corporate-image points of view, the severity of this type of issue is indeed high. But the severity field doesn't allow us to express that properly. So the need for the priority field becomes apparent. The priority field does allow product management to define this issue as high priority, but this creates the case where something is low severity but high priority.

**Instructor Notes:****1.5: Defect Reports**  
**Example on Severity & Priority**

Let's consider another case:

The anomalous server crash. We've all seen this type of issue. A server crash that occurs on the first full moon of every leap year but that is not reproducible by any human means on a consistent basis.

So how would this issue be categorized within the defect tracking system?

Well, since it is a server crash, many would argue it should be a high-severity issue. After all, the system is inoperable until the server is restarted. But what is the impact to the customer? In this case, the impact is quite small. Since the customer may never see this issue present itself at all in a production environment, it would be given a low priority and high severity by Product Management

**Instructor Notes:****1.6 Defect Report  
Users**

- Management
- Maintenance Team lead
- Maintenance Engineer
- Testing Team Lead

**Instructor Notes:****1.6 Defect Report  
Benefits for Maintenance Team Leads**

Allocate the Defect to the appropriate team member as soon as possible

- Quickly understand the software version and component responsible for issue

Effectively prioritize defect for fixing

- Is it halting the testing process?
- Are other functionalities dependent on this?
- Is it important under part release?
- Expected fixing date

To accurately take corrective and preventive actions for future developments

- Category wise, severity wise Defect status (functionality, modules, layers)
- Average turn around time for Defect fixing
- Defect density

**Instructor Notes:****1.6 Defect Report  
Benefits for Maintenance Engineer**

Identify the application version (mainly for products)

- Isolate the application version when multiple versions are being maintained

Quickly get to the root cause

- Concentrate not on symptoms but root cause to isolate the component creating issue

Know reproducibility and environment/situation of reproducibility

- Else do not waste time, arrange for necessary dependencies/settings

Analyze the log file and get clear understanding about the issue

- Check the exact details - data entered, actions taken, results generated, tables updated (application log, database log)

Contact the tester who found the Defect

- To get a first hand information & clarification directly and quickly

**Instructor Notes:****1.6 Defect Report  
Benefits for Testing Team Lead**

Plan retesting efforts

- Tentative dates when defects are expected to be fixed
- Estimated defects

Analyze quality of Defect reporting process

- Defect Acceptance Rate, Defect Communication Effectiveness

Increase accuracy on future estimates

- Generate accurate summaries for status reporting
- Application module wise, Severity wise total/open defects
- Functionality wise, severity wise total/open defect counts

Monitor performance of the testers

- Tester wise metrics

**Instructor Notes:****1.6 Defect Report  
Benefits for the Management**

To know the status of the Defects

- Application module wise, severity wise defect summary of open and closed defects
- Category wise severity wise open and closed defects
- Expected dates for fixing and closing of high severity defects

To analyze the performance of the teams

- Team wise – team member wise metrics – count, productivity

Effective follow-up

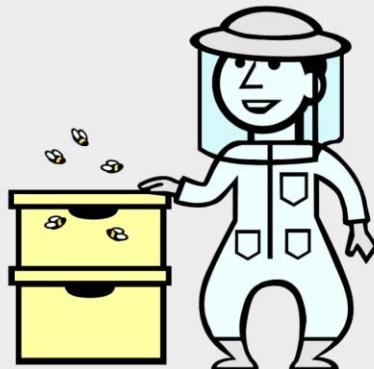
- Generate exception reports – Actions due in next n days, Actions pending for more than n days
- To take go/no go decision for next cycle/phase/production
- Defect summary in conjunction with Test Case execution summary

To know the risks involved

- Summary and details of known defects (with impact)

**Instructor Notes:****1.7 Defect Management  
Logging, Tracking & Analysis**

A Good Defect Management process helps to gather and manage information during each defect workflow i.e. from initial discovery to final resolution or deferral.



**Instructor Notes:****1.7 Defect Management  
Logging Defect**

Get empty defect template

Specify available information; keep updating as information becomes available

Categories of Defect Information

- General information
- Defect detection information
- Resolution information
- Status information



**Instructor Notes:****1.7 Defect Management**  
**Logging Defect – General Information**

Basic information on nature of defect, its repair priority, etc. :

- Description - Brief text
- Priority
- Severity
- Cause keywords (For further analysis)
- Symptoms (Database corruption, visible data wrong, cosmetic etc.)
- Phase found in
- Date reported
- Actual date of closure



**Instructor Notes:****1.7 Defect Management**  
**Logging Defect – Defect Detection Information**

Specify information about testing data related to defect, environment who found it etc.

Description

Build, log, cycle, procedure, case in which defect was found

Reported By - Name, Company

Hardware, software - Platform on which defect found

Attached Information

Additional Information



**Instructor Notes:****1.7 Defect Management**  
**Logging Defect – Resolution Information (Developer)**

How the defect was resolved

- Resolution from a codified list
- Fixed in build / version
- Resolution description
- Modified software - components modified to resolve defect
- Additional information
- Additional attachments

**Instructor Notes:****1.7 Defect Management**  
**Logging Defect – Status Information**

Current status of a defect and status history

- Description
- Status history with rows containing
  - Date
  - Action
  - Performed by
  - Actual effort
  - Resulting status
  - Next step assigned to
  - Estimated effort for next step
  - Expected date of completion



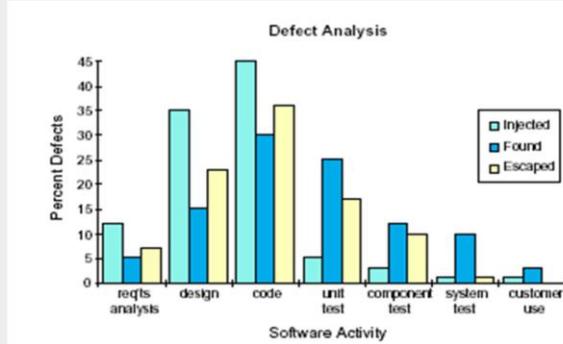
**Instructor Notes:****1.7 Defect Management  
Tracking Defect**

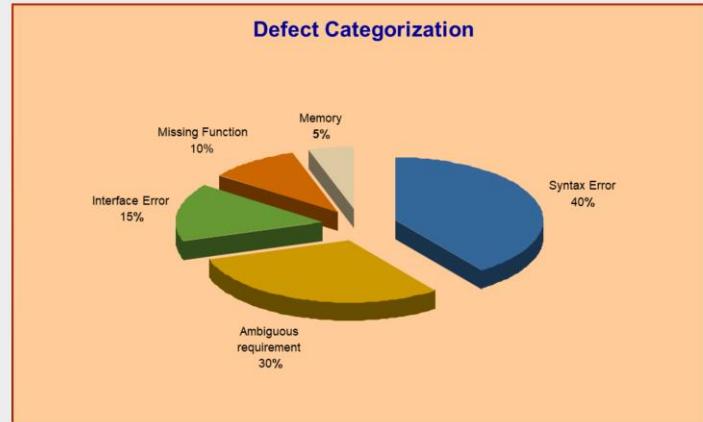
Summary Information on defects not yet closed

- Defect Identification (name / number)
- Date Reported
- Expected date of closure
- Severity
- Priority
- Current state
- Assigned to
- Selection by "Assigned to", "Priority", or "severity", etc.
- Sorting by various orders.

**Instructor Notes:****1.7 Defect Management  
Defect Analysis (Cont.)**

This chart shows the % defects found during each testing phase. Helps in analysis of the effectiveness of different phases in SDLC.



**Instructor Notes:****1.7 Defect Management  
Defect Analysis**

**Instructor Notes:****1.8 Defective Reports**  
**What is Defective Defect Report?****Defective defect Report**

- The inaccurate, incomplete and unclear defects results into defective defect report
- Impact of Defective defect report
- Wastage of time that is precious in tight schedule
- Inaccurate / incomplete status leading to wrong / no decisions
- Inaccurate statistics leading to inaccurate corrective / preventive measures
- Frustration and ill feeling between development and testing teams

**Instructor Notes:**

### 1.8 Defective Reports

#### The reasons for defective reports

Cannot reproduce

- If the maintenance engineer is not able to reproduce the bug, by using steps mentioned in defect report
- Already reported (Duplicate)
- Functionality is as per requirement
- Some one else is responsible
- Additional information needed – Error message detail, Data input, options selected, previous tasks executed etc
- Details provided are not clear
- Some attributes are not provided or not correct – Severity, Transaction Id, version, category etc
- Is a new requirement or change in requirement

For all above reasons the developer will change the status of the bug as rejected.

**Instructor Notes:**

### 1.8 Defective Reports

#### Root causes for defective reports

- An Assumption - Developer should be able to understand defect quickly and easily with little hint
- Providing all the steps, test data etc. takes lot of time to report
- Testers do not know the importance (usage) of details other than defect description
- Providing evidences for the defect are not considered important
- Informal communication process - through emails, verbal, and historical details are not maintained
- Some defects gets unknowingly fixed due to fixing of other defects
- Features of tool, process not known

**Instructor Notes:****1.8 Defective Reports  
Other Influencing Factors**

- An Requirement of large testing team size
- Large and Complex application architecture with Involvement of multiple development teams
- Teams working from different countries in different time zones
- Business users/testers and developers speaking different languages

**Instructor Notes:****1.9 Severe Defects**  
**The Most Severe Defect**

Wrong Severity is a severe defect

- Importance for go / no go decisions
- Importance for deferring the defect-fix to next release
- Credibility of development team is based on such defects

**Instructor Notes:****1.10 Defective Reports  
Certain Facts**

Quick fixing of some Defects would help first the testers themselves

- Show stoppers
- Dependencies and pre-requisites

Developers are human beings

- There are bound to be Defects in application

Varied interests and expectations – High defect count ->

- Good performance by test team but
- Bad performance by development team

The maintenance engineer is not the only user of the Defect report

- Maintenance team lead, Test team lead, Management

**Instructor Notes:**

### 1.11 Defect Free Reports

#### Importance of Effective Defect Reporting

From the Development Perspective

- Real Defects in the system/program
- Clear but brief information about the bug
- Steps to easily reproduce the problem
- Proper description of the problem if they happened to be more general
- Developer should be able to isolate the problem reported
- Increased productivity – in fixing the problem with least amount of effort
- Expect reports that convey the proper message and simplifies the process



**Instructor Notes:**

### 1.11 Defect Free Reports

#### Importance of Effective Defect Reporting

From the Test Perspective

- Reduce the defect life cycle
- Ensure that the defects get fixed by developers in the agreed timeframe
- Improve the credibility of the test
- Enhance teamwork between development and test
- Get better response from the development team
- Reduce things like "Need more feedback", "Works fine on my machine"

**Instructor Notes:****1.11 Defect Free Reports**  
**Importance of Effective Defect Reporting**

From the Management Perspective

- Improve productivity
- Get accurate information on the defects reported
- Reduce the time to market
- Get the correct metrics
- Take proper actions in timely resolution of the defects



**Instructor Notes:****1.12 Defect Free Reports  
Recommendations**

Increasing Awareness / Being conscious of

- Some realities in testing process
- Users of Defect report
- Importance of Defect report attributes to different users

Following process and guidelines

Preparation before starting the assignment

- Verify before recording defect
- Review Defect report before submitting

Use of tool with required features

Institutionalizing improvements

**Instructor Notes:****1.13 Writing Defect Free Reports**  
**Preparation**

Get complete understanding of Defect Tracking process and tool

- Guidelines for reporting, checking for duplicate defects, pre-defined definitions for severity, priority, categories etc

Establish / understand communication protocol

- Abbreviations and symbols, Providing references to documents, using standards

Acquire communication skill

- To provide clear, complete yet concise information about the defect

**Instructor Notes:****1.13 Defects  
Reporting and Communication Process**

Checking before Recording the defect

- It is really a Defect
- It is not yet reported
- It is a specific or a general issue

**Instructor Notes:****1.13 Defects  
Reporting and Communication Process (contd.)****Finding and reporting**

- The root problem
- The shortest way to recreate the Defect
- All the other information that can help
- All evidences – screen shots, references to other documents, correspondences etc

**Ensuring**

- CAN PIG RIDE??

When in doubt report the defect - Primary objective is to get as less defects (if possible no defects) in production and not high Defect Acceptance Rate

**Instructor Notes:****1.13 Defects  
Reporting and Communication Process (contd.)**

- Condense
- Accurate
- Neutralize
- Precise
- Isolate
- Generalize
- Recreate
- Impact
- Debug
- Evidence

Key points to make sure the next defect report you write is an effective one.

1. **Condense** - Say it clearly but briefly
2. **Accurate** - Is it a defect or could it be user error, misunderstanding, etc.?
3. **Neutralize** - Just the facts. No zingers. No humor. No emotion.
4. **Precise** - Explicitly, what is the problem?
5. **Isolate** - What has been done to isolate the problem?
6. **Generalize** - What has been done to understand how general the problem is?
7. **Re-create** - What are the essentials in triggering/re-creating this problem?  
(environment, steps, conditions)
8. **Impact** - What is the impact to the customer? What is the impact to test?
9. **Debug** - What does development need to make it easier to debug?  
(traces, dumps, logs, immediate access, etc.)
10. **Evidence** - What documentation will prove the existence of the error?

It is key to make sure that you have covered the essential items that will be of most benefit to the users of the defect report.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Condense - Example**

Best Practice	Defect Remark
<b>Don't:</b> Suffers from too much information, most of which is not helpful.	I was setting up the test whose real intent was to detect memory errors. In the process, I noticed a new GUI field that I was not familiar with. I decided to exercise the new field. I tried many boundary and error conditions that worked just fine. I cleared the field and attempted to advance to the next screen, then the program abended. Several retries revealed that anytime, there is not any data for the "product description" field, you cannot advance to the next screen or even exit or cancel without abending.
<b>Do:</b>	The "exit", "Next" or "cancel" functions for the "product information" screen abends when the "product description" field is empty or blank.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Accurate - Example**

Be extremely sure that what you are reporting is really a bug

Don't lose credibility by reporting wrong Defects

Do your homework before you write a problem

Check with developer or Senior tester before reporting

Ask questions like,

- Is there something in the setup that could have caused this?
- Could this be a result of network or environmental problem?
- Do you really understand how this is suppose to work?

Check with developer or senior tester before reporting.

Don't be afraid to report problems. Do your best to ensure they are valid problems.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Neutralize - Example**

Best Practice	Defect Remark
<b>Don't:</b> The first clause may be interpreted as a jab at the developer and adds no useful information.	As could have been determined from the original defect, with very little effort, function ABC does indeed abend with any negative value as input.
<b>Do:</b>	Function ABC abends with any negative value. For example: -7, -1, -32767.

1. State the problem Objectively
2. Don't use humor or emotionally charged zingers.
3. Even if defects are rejected provide more information which will be helpful to the developer. This will help in credibility in the long run.  
Even if defects are returned back provide more information that will be helpful to the developer. This added bit of professionalism will give you respect and credibility in the long run
4. What you think funny may not be interpreted funny by the developer who is working overtime and is stressed with deadlines. They just create barriers to communication and teamwork.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Precise - Example**

Best Practice	Defect Remark
<b>Don't:</b> In this example, it is hard to tell if the problem is (1) the twinmax port not working or (2) printer not returning to ready	Issuing a cancel print when job is in PRT state (job is already in the printer and AS/400 is waiting to receive print complete from printer) cause the twinmax port to not time out. The printer never returns to ready state and indefinitely displays "Printing from Tray1" in the op-panel.
<b>Do:</b> Precede the description with the short summary of exactly what you perceive the problem to be.	Cancelling the job, while it is printing causes printer to hang.  Issuing a cancel print when job is in PRT state (job is already in the printer and AS/400 is waiting to receive print complete from printer) cause the twinmax port to not time out. The printer never returns to ready state and indefinitely displays "Printing from Tray1" in the op-panel.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Isolate - Example**

Invest a reasonable amount of effort in isolating a problem  
Try to find out shortest and easier steps to reproduce the problem  
Ask yourself if anything external to the code is causing the problem for e.g. network, etc.  
If doing an end-to-end testing can you specify which exact component is causing the failure  
For testing multiple input conditions vary the input condition until you find the value which triggered the problem  
Your ability to isolate in a large part defines your value-add as a tester  
e.g. You found a problem while printing a postscript document, even if you think the problem occurs while printing the postscript document, specify the exact document that you used for printing

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Generalize - Example**

Incorrect

- Date displayed on the Payment Screen is not in the format mm/dd/yy .

Correct

- Date displayed on the Payment Screen, Batch Screen, Billing Summary screen is not in the format, mm/dd/yy .

Fixes are made often as per the problem is reported.  
Identify if its a more general problem and needs a generic fix.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Recreate - Example**

If you can re-create the bug, you should explain clearly and concisely what is required to do the re-create.

List all steps, exact syntax, file name or sequences you encounter to re-create the problem.

If you find more reliable and shorter methods while verifying/re-creating document it.

If you are not able to re-create it or suspect about it, gather all the relevant information and pass it on to the developer to see if they want to examine the system

Incorrect

- Assume that the problem cannot be re-created if you haven't verified that it can be

Correct

- If you cannot or haven't re-created the problem it is important to note that in the defect remarks.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Impact - Example**

- What is the impact if the bug was to surface in the customer/production environment?
- If you think that the defect won't get sufficient priority then state the potential impact and sell the defect.
- Don't oversell, but make sure that the readers understand the probable impact on the customer.

**Example :**

1. System crashes when I hit the enter key.
2. You may notice a typo error on a window. It may be a very minor, but if that results to be an offensive word, then such typo errors need to be fixed.

**Instructor Notes:****1.13 Reporting and Communication of Defects**  
**Debug - Example**

- What will help the developer need to debug the problem? Are there any traces, dumps, logs, etc. that should be captured and made available along with the defect report.
- Provide correct pointers in the logs, dumps that will help the developer to resolve the defect as fast as possible.

**Instructor Notes:****1.13 Reporting and Communication of Defects  
Evidence - Example**

- What proves that the defect you report is an error? Have you provided both the expected and the actual results? Is there any documentation that supports your expected result?
- Evidence may take the form of user guides, requirements, design, etc.
- It also may be past comments from customers, de-facto standards, competing products
- Don't assume everyone see things same as you do.
- Don't assume that 3 weeks from now you will remember why this was a bug
- Provide even more evidence when you think this situation may not be accepted as a bug.

**Instructor Notes:****1.14 Writing Effective Defect Reports Guidelines**

When you file a defect, it needs to be easily conveyed to the developer

You must provide clear information

- Fill out as many fields as you can
- Provide screen shots, log files, URL's and references to similar defects
- Include detailed steps to reproduce the issue

**Some more points to remember**

- Refer to requirements or design documents where appropriate.
- When possible, check to see if the problem occurred in an earlier build.
- Include data like username, date, time, and details that make the problem easier to find
- If you checked a log file and find nothing, include a comment like "Nothing in [log name].log."
- You may want to try other scenarios and include the results.

**Example of good Description:**

*On the bills page, some amounts have no decimal after the amount, others have two. Since this is currency, we should be consistent and always display two decimals (or whatever is standard) after the amount - even if the last digit is a zero.*

**Example of a bad description:**

*went to page. decimals aren't right*

**Instructor Notes:****1.15 Using Tools for Reporting  
Advantages****With Primary features**

- Built in validation checks
- Maintenance of history
- Generation of summarized information, metrics
- Ability to quickly search on specified criteria
- Multiple attachments
- Reduced other communication issues
- Online status

**With other optional features**

- Proactive notification when no action taken on due date
- Involvement of translator for translation when needed
- Understand number of items on which actions to be taken
- Change Request approval process

**Instructor Notes:****1.16 Project/Organization Level Process**  
**Project/Organization Level Process**

Competency Development of test team in

- Communication skills
- Defect management

Project specific familiarization process

- Communication protocol
- Defect reporting and tracking process
- Defect tracking tool

Summary / Metrics generation, sharing and monitoring

- Defect Acceptance Rate
- Defect Communication Effectiveness
  - (Total Defects reported / Number of times Defects are communicated to Maintenance team) \* 100

**Instructor Notes:****1.17 Defect Free Reports**  
**Advantages**

- Improved project control
- Improved quality of report
- Improved productivity
- Improved cycle time
- Reduction in overall effort
- On time delivery
- Overall satisfaction
- Improved predictability

**Instructor Notes:****1.18 Nine Commitments****Nine Commitments worth making to developers**

We'll test your code as soon as we can after it's built.

We'll test important things first, and focus on important problems.

We'll write clear, thoughtful, and respectful problem reports.

We'll try not to be a bottleneck for development.

We'll tell you how we're testing, and consider your suggestions.

We'll look for ways to test better and faster.

We will not waste your time.

We will create order out of chaos.

We will always remember that collectively we win or loose as a team.

**Instructor Notes:****Summary**

In this lesson, you have learnt:

- Defect report is an important deliverable since it gets referred by maintenance team, testing team, management
- The inaccurate, incomplete and unclear defects results into wrong decisions
- Follow process and guidelines
- People fixing defects are most likely to be different than original developers
- Institutionalizing process and building competencies for defect free defect reporting

**Summary**

**Instructor Notes:**

Answers:

Question1: Option4

Question2: False

Question3: True

**Review Question**

Question1: Which of the following will be entered by test engineer in the defect report

- Option 1: Reported By
- Option 2: Resolution Details
- Option 3: References
- Option 4: All of the above

Question 2: Before you log a defect it is not necessary to verify whether it is duplicate because it is time consuming.

- True/ False

Question 3: Adding attachments is easy if we are using any tool to log the defect

- True/ False