What is the Test Report?

Test Report is a document which contains

- A **summary** of test activities and final test results
- An **assessment** of how well the Testing is performed

Based on the test report, the stakeholders can

- **Evaluate** the **quality** of the tested product
- Make a **decision** on the software release. For example, if the test report informs that there're many defects remaining in the product, the stakeholder can delay the release until all the defects are fixed.

Test Report Example

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					**130**

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

Figure 6.1

Why Test Report?

The following scenario will show you why we do need the Test Report

Do you know the root cause of this problem? Why does the website still has defects even when your Team has already tested it?

The problem is you ignored the reporting & evaluation phase in Test Management. The boss has no information to evaluate the quality of this website. They just trusted what you said and released the website without knowing its testing performance.

The typical benefits of a test report include:

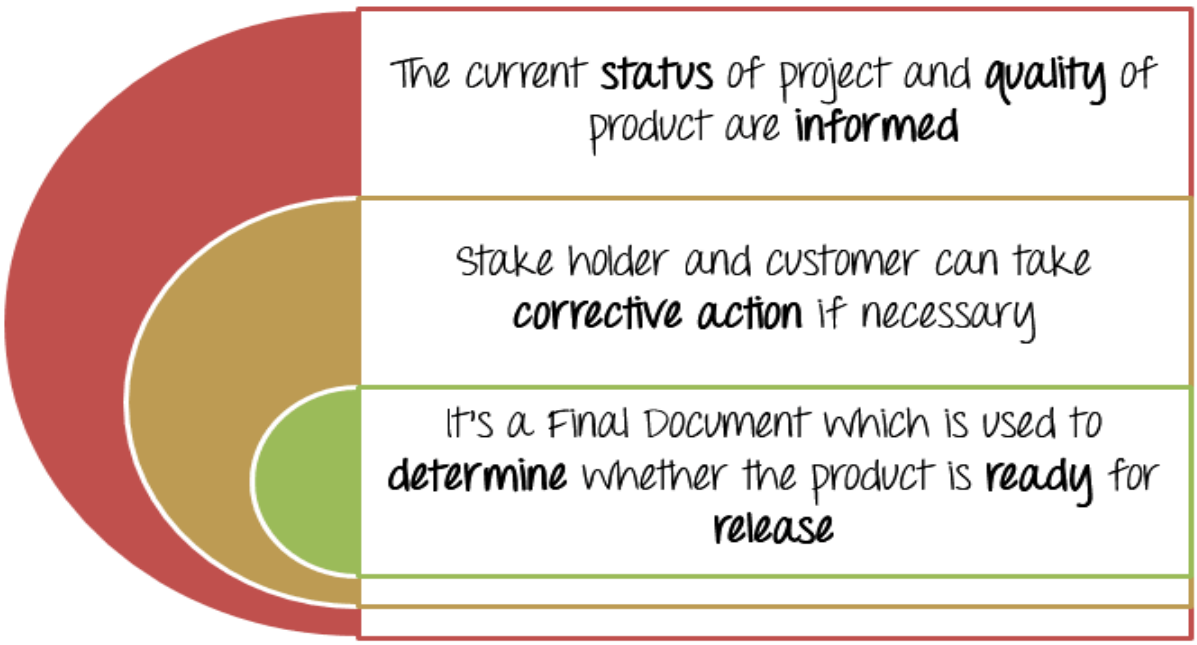

Figure 6.2

How to make a good Test Report?

To answer this, you must know -

What does a test report contain?

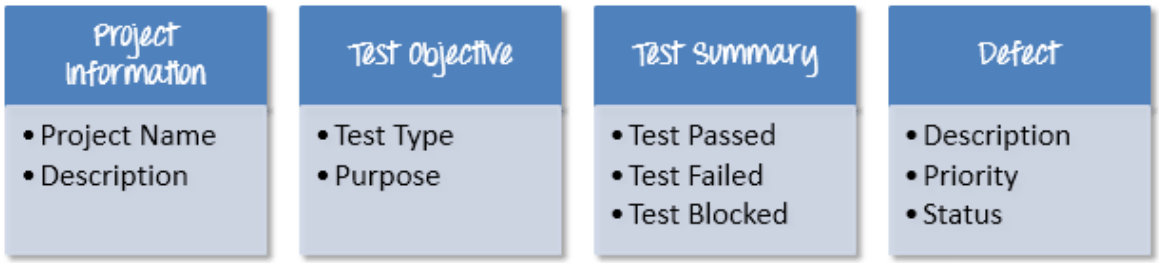

Figure 6.3

Project Information

All information of the project such as the project name, product name, and version should be described in the test report.

For example, the information of Guru99Bank project will be as follows

Project Overview				
PROJECT BASIC INFORMATION				
Project Name	Guru99 Bank			
Name of product (Product Number)	Banking website www.demo.guru99.com			
Product Description	The banking website			
Project Description	**<Mision of project>** Conduct testing to verify the quality of this website Ensure the website is released without any defects **<Project's output product>** Test Summary Report & Evaluation			
	Project Type	Testing/Verification		
Project Duration	Start date	10/1/2013	End date	10/31/2013

Figure 6.4

III. Reporting What You Find:

2. Getting Your Bugs Fixed (Question: Explain the steps to fix the bug in software testing. = 8 marks)

1. There's not enough time. Every project always has too many software features, too few people to code and test them, and not enough room left in the schedule to finish.

2. It's really not a bug: Maybe you've heard the phrase, "It's not a bug, and it's a feature!" It's not uncommon for misunderstandings, test errors, or spec changes to result in would-be bugs being dismissed as features.

3. It's too risky to fix: Unfortunately, this is all too often true. Software is fragile, intertwined, and sometimes like spaghetti. You might make a bug fix that causes other bugs to appear. Under the pressure to release a product under a tight schedule, it might be too risky to change the software. It may be better to leave in the known bug to avoid the risk of creating new, unknown ones.

4. It's just not worth it: This may sound harsh, but it's reality. Bugs that would occur infrequently or appear in little-used features may be dismissed. Bugs that have work around, ways that a user can prevent or avoid the bug, often aren't fixed.

5. It all comes down to a business decision based on risk. One more item should be added to this list that can often be the contributing reason for all of them:

- **Bugs are reported ineffectively**: The tester didn't make a strong enough case that a particular bug should be fixed. As a result, the bug was misunderstood as not being a bug, was deemed not important enough to delay the product, was thought to be too risky to fix, or was just plain considered to not be worth fixing.
- **Report bugs as soon as possible**: The earlier you find a bug, the more time that remains in the schedule to get it fixed. If the same bug is found a few hours before the release, odds are it won't be fixed. Figure 6.5 shows this relationship between time and bug fixing on a graph.

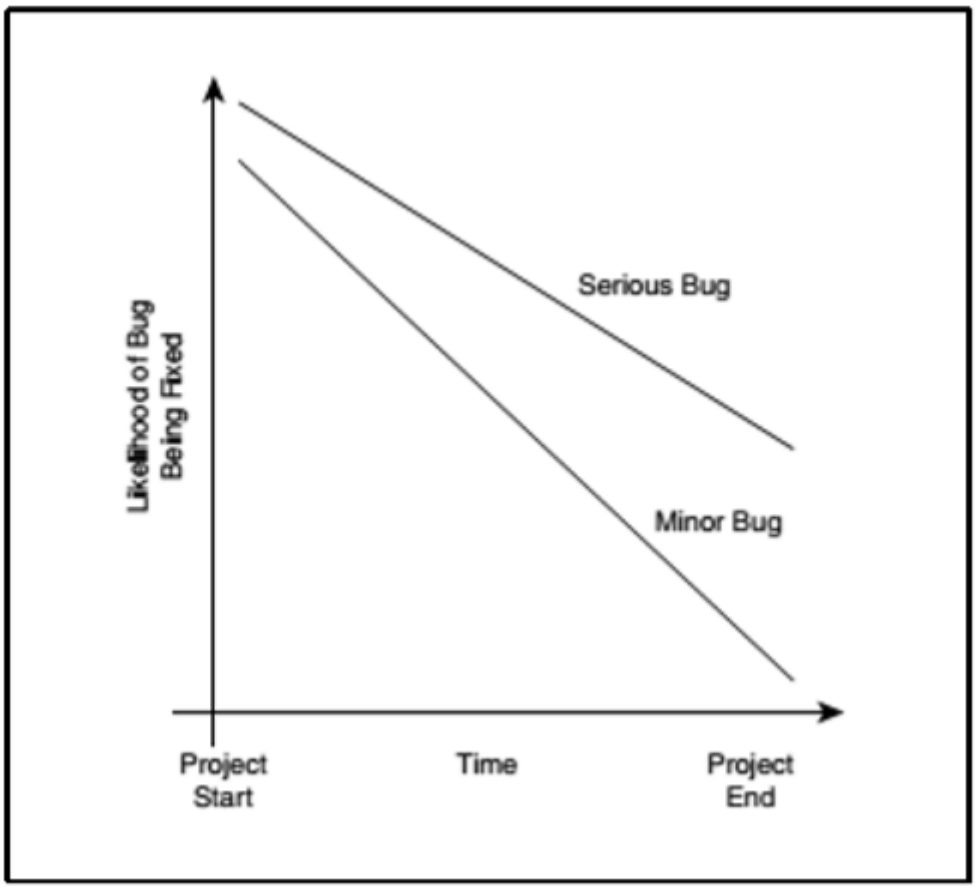

Figure 6.5

3. Isolating and Reproducing Bugs

1. Isolating and reproducing bugs is where you get to put on your detective hat and try to figure out exactly what the steps are to narrow down the problem.

2. The good news is that there's no such thing as a random software bug—if you create the exact same situation with the exact same inputs, the bug will reoccur.

3. The bad news is that identifying and setting up that exact situation and the exact same inputs can be tricky and time consuming. Once you know the answer, it looks easy. When you don't know the answer, it looks hard.

4. Reporting What You Find.

4. Not All Bugs are Created Equal

1. The following list of common classification of severity and priority should help you better understand the difference between the two.

2. Keep in mind, these are just examples. Some companies use up to ten levels and others use just three.

3. No matter how many levels are used, though, the goals are the same.

Severity

- System crash, data loss, data corruption
- Operational error, wrong result, loss of functionality
- Minor problem, misspelling, UI layout, rare occurrence

Priority

- Immediate fix, blocks further testing, very visible
- Must fix before the product is released
- Should fix if time permits.
- Would like fix but can be released as is.

5. A Bug's Life Cycle (Question: Explain the principle attributes of tools and automation in software testing = 4 marks)

The term life cycle refers to the various stages that an insect assumes over its life.

1. When a bug is first found by a software tester, it's logged and assigned to a programmer to be fixed.

2. This state is called the open state.

3. Once the programmer fixes the code, he assigns it back to the tester and the bug enters the resolved state.

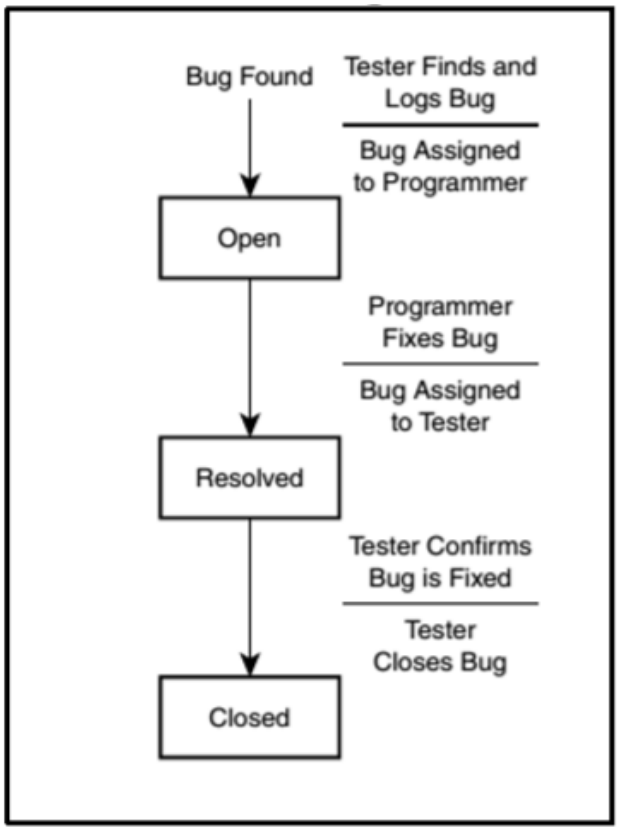

Figure 6.6

4. The tester then performs a regression test to confirm that the bug is indeed fixed and, if it is, closes it out.

5. The bug then enters its final state, the closed state.

6. Bug Tracking System:

b) The Standard

1. It should be clear that the bug-reporting process is a complex beast that requires a great deal of information, a high level of detail, and a fair amount discipline to be effective.

2. Everything you've learned so far in this chapter sounds good on the surface, but to put it into practice requires some type of system that allows you to log the bugs you find and monitor them throughout their life cycle.

3. A bug-tracking system does just that.

c) The Test Incident Report Q.7

1. Reviewing the standard is a good way to distil what you've learned about the bug-reporting process so far and to see it all put into one place.

2. The following list shows the areas that the standard defines, adapted and updated a bit, to reflect more current terminology.

- **Identifier**: Specifies an ID that's unique to this bug report that can be used to locate and refer to it.
- **Summary**: Summarizes the bug into a short, concise statement of fact. References to the software being tested and its version, the associated test procedure, test case, and the test spec should also be included.
- **Incident Description**: Provides a detailed description of the bug with the following information: Date and time Tester's name Hardware and software configuration used Inputs Procedure steps Expected results Actual results
- **Impact**: The severity and priority as well as an indication of impact to the test plan, test specs, test procedures, and test cases.

d) Manual Bug Reporting and Tracking

1. The 829 standard doesn't define the format that the bug report should take, but it does give an example of a simple document.
2. Figure 6.7 shows what such a paper bug report can look like.

WIDGETS SOFTWARE INC. **BUG REPORT** BUG#:_____

SOFTWARE:_____ RELEASE:_____ VERSION:_____

TESTER:_____ DATE:_____ ASSIGNED TO:_____

SEVERITY: 1 2 3 4 PRIORITY: 1 2 3 4 REPRODUCIBLE: Y N

TITLE:_____

DESCRIPTION:_____

RESOLUTION: FIXED DUPLICATE NO-REPRO CAN'T FIX DEFERRED WON'T FIX

DATE RESOLVED:_____ RESOLVED BY:_____ VERSION:_____

RESOLUTION COMMENT:_____

RETESTED BY:_____ VERSION TESTED:_____ DATE TESTED:_____

RETEST COMMENT:_____

SIGNATURES:

ORIGINATOR:_____ TESTER:_____

PROGRAMMER:_____ PROJECT MANAGER:_____

MARKETING:_____ PRODUCT SUPPORT:_____

Figure 6.7

e) Automated Bug Reporting and Tracking

1. Once a bug is entered, and really anytime during its life cycle, new information may need to be added to clarify the description, change the priority or severity, or make other minor tweaks to the data.
2. That this dialog box provides additional data fields over what the new bug window provided.
3. Editing a bug allows you to relate this bug to another one if you find one that seems similar.
4. A programmer can add information about how much progress is made in fixing the bug and how much longer it will take.
5. There's even a field that can put the bug "on hold," sort of freezing it in its current state in the life cycle.