1,2,3… Software Testing…
1,2,3…

(An Introduction into Software Testing)

By Sameer Nathan
Epiance Inc.
Software Testing

PROLOGUE: .................................................................................................................. 3
WHY TESTING: .................................................................................................................. 4
INTRO INTO TESTING: .................................................................................................... 4
  BASE MODEL FOR TESTING PROCESS: .......................................................... 5
  QA: ................................................................................................................................. 7
  QC: ................................................................................................................................. 7
WHY DO SOFTWARE HAVE BUGS: ........................................................................ 8
TESTING PROCESS: ......................................................................................................... 10
  TEST CASES: .................................................................................................................. 10
  DEFECTS - WHAT THEY ARE AND WHAT TO DO WITH THEM: ................................ 10
  TEST CYCLE: ................................................................................................................. 11
DIFFERENT KINDS OF TESTING: .................................................................................. 11
  BLACK BOX TESTING: ................................................................................................. 11
  WHITE BOX TESTING: ................................................................................................. 13
TOOLS THAT HELP IN TESTING: .................................................................................. 14
  AUTOMATION TESTING: ............................................................................................... 15
  PERFORMANCE TESTING: ............................................................................................ 15
  CODE COVERAGE: ......................................................................................................... 16
  MEMORY LEAKAGE: ..................................................................................................... 16
  LOAD TESTING: ............................................................................................................. 17
BASIC INTRODUCTION TO ACTUAL TESTING AND IT’S IMPLICATIONS: .................. 18
  PRODUCT TESTING: ...................................................................................................... 18
  INSTALLATION TESTING: ............................................................................................ 20
TESTING: PRODUCT COMPANY VS. PROJECT COMPANY: ...................................... 23
BASIC SKILLS REQUIRED BY A TESTER: ................................................................. 24
  CASE STUDY OF PROBLEMS CAUSED INCASE TESTING NOT DONE PROPERLY: (ALL CASE
  STUDY INFORMATION FROM WORLD WIDE WEB) ..................................................... 24
**Prologue:**

Who would want to make products of no standards or stability in today’s IT world? Well, if there are any, then they are the ones who do not consider Software Testing as a must. They also do not consider the quality of the product.

Software Testing, that define Software Quality, is of major importance to all the software companies in today’s highly competitive filed. Yes, a developer can develop products and check the functionalities of the product, but he can never certify the quality and stability of the product. Especially, after the integration of his modules with those of his co-workers.

A product can be made better (or at it’s best) only after it goes through a proper process of development, testing and bug fixing. Not only should it go through this process, but there should also be a good and healthy relation between the developers and the testers to attain this objective.
Why Testing:

What do you do first when you go out to buy vegetables or dress? We first check the quality of the vegetable or the quality of the cloth, then we go ahead and see whether there are any smashed area in the vegetable or whether there are any stitches loose in the dress and things like that. The same thing is applied to software in a different manner.

Firstly it is seen whether the product is of any use to the client. Then the product is checked to see whether it serves the purpose and whether the quality of the product (in-terms of performance, stability, error handling etc) is good. Other factors like the cost and license comes later on. But this initial check is what makes the most towards a successful deal, like they say, “The first impression is the best impression”.

No Organization would want to create a Bad Reputation as a result of their product being buggy. Apart from finding bugs, testing can also make sure that the Product is as per the specifications. It also helps in modifying features to make it more Usable and Friendly.

Now who can confirm whether the product is as per the specifications and whether it is stable and certify the quality of the product? Can the Developers who have developed the product certify? Can the Management who checks only the base level functionality certify? Can the Sales people certify? Only a specific group who has used the product in a large scale and who knows the limitations and problems that can be faced, can to some extend certify the products quality. This group of people comprises the TESTING department. They are the people who can say what will completely work and at what all bizarre scenarios can problems arise. They will be able to tell whether the base functionality works fine for most of the normal usage scenarios.

Intro into Testing:
Software Testing involves the entire software development PROCESS - monitoring and improving the process, making sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with.

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn’t or things don’t happen when they should. In short it can be termed as “Creatively Destroying the Product”.

One of the major factors that the tester has to keep in mind is, “Check the quality of the PRODUCT and not the quality of the PROGRAMMER”. His work is to find problems that can arise using the functionalities of the product, and not in analyzing how good the programmer has developed the product. If he does his work keeping the other factor in mind, the situation is just going to clash in a major way, resulting in a catastrophe within the organization.

Base Model for Testing Process:

There are a lot of methods and standards set for testing software. It is upto the company or the Testing department after evaluating different methods to decide which of the approach to take up. You could also create methods as per your specifications and implement it. It is not said anywhere that you have to use only these methods. Each of the testing process or model is based on certain factors and flow, which can be summarized in the below figure.
The basic flow of the process of testing can be derived from the above figure as: Development develops the Product and sends it to Testing Department. Testing Department tests the products and gathers the Defects. The Defects found out is communicated to the Developers. The Defects are fixed by the developers and sent back for testing. The above cycle is repeated from 2 to 4 until the management decides that the critical defects have been completely fixed.

But like said earlier, it is not mentioned anywhere that the method has to be followed like this only. Sometimes Testers are also called in during the design phase of the product so that logical problems are solved at the design time itself.

Then again there are the terminology of QA and QC running around in the Software Testing cycle. They can be described as done below:
QA:

QA stands for Quality Assurance. It is concerned more with being able to distinguish between those activities that modify the development processes to prevent the introduction of flaws. Software QA involves the entire software development PROCESS – monitoring and improving the process, making sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with. It is oriented to 'prevention'.

QC:

QC stands for Quality Control. Here the Organizations focus is on testing a group of quality-related attributes such as correctness, security, portability, inter-operability, usability, and maintainability. Organizations use statistical sampling, measurements of confidence levels, trustworthiness, and reliability goals to drive the testing process.

The topics talked here are mainly to do with QC.
Why Do Software have Bugs:

1. **Miscommunication or no communication** - as to specifics of what an application should or shouldn’t do (the application’s requirements).
2. **Software complexity** - the complexity of current software applications can be difficult to comprehend for anyone without experience in modern-day software development.
3. **Programming errors** - programmers, like anyone else, can make mistakes.
4. **Changing requirements** - the customer may not understand the effects of changes, or may understand and request them anyway - redesign, rescheduling of engineers, effects on other projects, work already completed that may have to be redone or thrown out, hardware requirements that may be affected, etc.
5. **Time pressures** - scheduling of software projects is difficult at best, often requiring a lot of guesswork. When deadlines loom and the crunch comes, mistakes will be made.
6. **Egos** - people prefer to say things like:
   - ‘no problem’
   - ‘piece of cake’
   - ‘I can whip that out in a few hours’
   - ‘it should be easy to update that old code’

   instead of:
   - that adds a lot of complexity and we could end up making a lot of mistakes’
   - ‘we have no idea if we can do that; we’ll wing it’
   - ‘I can’t estimate how long it will take, until I take a close look at it’
   - ‘we can’t figure out what that old spaghetti code did in the first place’

   If there are too many unrealistic ‘no problem’s’, the result is bugs.
7. Poorly documented code – it’s tough to maintain and modify code that is badly written or poorly documented; the result is bugs. Especially when someone else handles it from the Originator of the code.
Testing Process:

The testing process involves confirmation that the product has been developed as per the specifications (Functional Spec) set earlier, and that the basic functionality of all the features implemented works properly. To achieve this Testers usually run what is called Test Cases, which is created to cover the different features and functionalities of the product.

Test Cases:
The Test Cases are generated during the development time of the Product. The main aim of these documents is to cover the basic functionalities of the Product using normal user scenarios. Which means that they are generated from the Functional Spec. Test Cases are modified during the process of the Product Development as and when new features are added or when the existing features are modified.

For example:
Test Cases generated to test the Open feature of word would be something like:
Test Case 1: Click on File --> Click on Open
Test Case 2: Press Ctrl+O.
Test Case 3: Press Alt+F and then Alt+O.
Test Case 4: Press Ctrl+F12.

Note that Test Cases can be in any format and not necessary like mentioned above.

Defects – What they are and what to do with them:
Defects are scenarios that cause the Product to throw errors or cause the Feature not to function as mentioned in the Functional Spec. Running the Test Cases will come out with certain defects. Exploratory Testing (testing off hand without any documents) will also unfold a lot of defects. Performance of the product is also tested and degradation in it is also treated as a defect.
Testers need to note down the Feature and the scenario that causes the problem and duly communicate the same with the Project Manager or Developer. The mode of communication depends on the process that the Organization has decided to implement. It could be through a Database, Mails, Documents or any other method. This process is called Defect Communication or Defect Reporting. The main aim here is to communicate these issues and have them resolved. Testers also need to keep track of the defect to confirm that they have been fixed and not faced there after.

Once these defects have been fixed, the Testers verify it, i.e. they confirm that the defect has been fixed, by running the same scenario again. This process of verifying whether the defects have been fixed is called Defect Verification. The whole process of finding defects, and verifying defects is considered as a Test Cycle.

Test Cycle:
A Test Cycle is the period in which the Product is tested and defects are verified. Before the release of a product, it will have gone through numerous Test Cycles to confirm the Quality of the product.

Different Kinds of Testing:

Testing can be conducted in different kinds like Manual Testing, Automation Testing, Memory Testing, Performance Testing, Usability Testing, etc. The different kinds of testing are again broadly classified into two major sectors called the “Black Box Testing” and the “White Box Testing”. We shall look into some of the commonly used testing methodologies of White Box testing and Black Box testing here.

Black Box Testing:
Black Box testing is the testing all the features and functionalities of the final product. It is also called as Closed Box Testing. The Tester has no information about the internal architecture or functional mechanism of the product. Black Box testing is a wide range and has many sub kinds in it.
Acceptance Testing: When the product is delivered for testing, it is first checked whether the product is in a functional condition to test. This is called Acceptance Testing. The Testers can generate an Acceptance Test Case, which can be run for Acceptance Testing.

Ad Hoc Testing: This method is also called Exploratory Testing where in no Test Case designs are used to test the product. The Tester goes by his intuition and creativity in finding out problems with the product.

Automation Testing: This method of testing is done using third party tools. In this method, we automate, or run a particular Test Case for a specific number of time. This helps in finding out the consistent passing of the test case and also helps in finding our problems related to memory, Performance etc. More about this method is dealt in detail later.

Boundary Value Testing: Certain features will require inputs to be taken from the user, like Login name in a site. There will also be a lower end limit and higher end limit to the input value to be given. In this kind of testing, we check the feature using values just below or above the lower limit and the upper limit.

Compatibility Testing: Compatibility Testing confirms that the Product installed does not hinder the functionality of any other product installed on the machine. Products are said to be compatible with each other if they can share data between them or if they can simultaneously reside in the same computers memory. In short Compatibility testing checks that one product works with another.

Integration Testing: This testing is done when the different modules of the product have been integrated together. It is an orderly process, which is carried out until all the modules have been integrated into the system.

Manual Testing: That part of software testing that requires operator input, analysis, or evaluation.

Performance Testing: The tests might determine which modules execute most often or use the most computer time. Then those modules are re-examined and recorded to run more quickly.

Regression Testing: This process of testing is done when new features are added or when features have been modified to confirms that the features that have been functioning are still functioning and that defects which has been fixed still remain fixed.
**Usability Testing:** Usability testing is done to check the flow of the product and the user friendliness of the product. It helps in finding out whether the user is able to interact with the product and achieve his goal. People who are novice to the product usually do this kind of test, and the Testers and Developers study the user while he is doing the process.

**Installation Testing:** Here the installation media and different kinds of installation process on different Operating systems with different configurations are tested. What good is a product if it does not install properly on different computer configurations?

**White Box Testing:**
White box testing involves testing at the Code Level and is done usually at the coding stage. Here the inputs required for testing is fed to the program and the output is checked. The Tester does not know or pretends not to know how the program works. The are a lot of different kinds of White Box testing. Some of the major methods are mentioned below.

**Boundary Value Testing:** This method can be used as black box and as white box. The feature tested here is similar to Black Box testing as mentioned earlier. (See Black Box testing section)

**Branch Testing:** This testing is done to satisfy the coverage criteria such that for each decision point each possible branch be executed at least once. For example if the execution reaches a Case statement, this kind of testing should cover each of the Case conditions at least once.

**Top–Down and Bottom–Up Testing:** In the Top–Down testing the highest level modules are tested first whereas the Bottom–up testing, the lower modules are tested first, and then the higher level modules.

**Hybrid Testing:** A combination of top–down testing combined with bottom–up testing of prioritized or available components.

**Incremental Testing:** Under this each piece of a module is first tested separately. This testing makes it easy to pin down the cause of the error but to test it requires a special code. Each piece is individually tested with focus on it and is thoroughly tested.
**Special Case Testing:** This kind of testing can be conducted as both Black Box testing and White Box testing. Here testing is done by giving input values that seem likely to cause program errors like "0", "1", NULL, empty string.

**Statement Testing:** Testing that satisfies that each statement in a program is executed at least once during program testing. It is said, “An untested code is a hidden Bomb”. It is true in the sense that if it is not tested, we can never say whether it will function properly or throw up errors.

Some of the other testing methods and terminologies used in the Testing field are:
- Alpha Testing
- Beta Testing
- Assertion Testing
- Big Bang Testing
- Design-Based Testing
- Development Testing
- Error Based Testing
- Equivalence Testing, etc.

There are tools available in the market that helps in doing White Box testing and the different kinds of testing under White Box testing. Bug Trapper from Mutek Solutions is one of the applications that help in conducting White Box testing. This application traces the path of the execution and captures the bug along with the path of execution and the different input values that had resulted in the bug.

**Tools that help in Testing:**

Testing is a widely growing area in today’s IT world. The need to do Software testing has inspired lot of companies into developing tools that help in the process of Software testing. In today’s market, you can get tools that will help in conducting Automated Testing, Code Coverage,
Code Analysis, Memory Leakage, Performance Testing, Load Testing, Stress Testing, etc.

We shall look into some of these Testing areas and the tools used in conducting them. Note that there are lots of software available in the market apart from those mentioned here.

Automation Testing:
Automation testing has been one of the major developing areas in Software testing. In this process of testing, repeated tasks or vulnerable areas of the product are run again and again to make sure that they do not cause problems. It is a tedious task for the Tester to run the task n number of times.

Tools have been created to help the Testers perform this operation. Like Rational Robot from Rational Software. Microsoft Visual Test from Rational. Silk Test, Silk Pilot from Segue Software Systems.

JAVA based applications are also developed in full fledge and gaining popularity among the populations. For testing JAVA based applications, there are specific tools, like Jtest from ParaSoft.

Performance Testing:
Performance Testing is of great importance to any software. This is done to make sure that the product does not take up much of the system resource and time in executing it’s task. Imagine the reaction of a user incase the save operation takes up more than 5 minutes.

This method of testing can be treated as Black Box testing or as White Box testing depending on what is used at the testing level. If we use the code then it is White Box testing, if we use the product then it is Black Box testing.

To check the performance of a product, tools are available that note down the function being called and the time taken for it’s execution. When we
do the task repeatedly, the time taken for each execution at different module level is noted down and analyzed. Rational Quantify from Rational, Code Test from Applied Microsystems, are some of the Performance Test tools present in the market.

Code Coverage:
Code coverage is the process in which the Test Cases are compared with the code to confirm that all the statements in the code has been tested. Tools available help in this process by highlighting the statements that has been executed after running a test case. Code coverage falls under the White Box testing family as here the code is being analyzed after the execution of the Test Case.

Rational Purify from Rational Software, Code Test from Applied Microsystems are some of the tools, which can help, in this testing process.

This kind of testing should be conducted incase time permits because as a Tester, we can never afford to leave parts of the code untested.

Memory Leakage:
Running the product should not cause the system to go low on resources. The chances of the resource consumption on usage of product are high. This is because whenever the product is using a memory location, after the usage, it has to be released, which if not done results in resource consumption (called memory leakage), and we know that for a product to function, it will have to use memory at some point or the other.

Memory leakage can be tested at both code level and at product execution level, making it fall under both Black Box testing and White Box testing. At the Black Box testing level, the Tester has to decide whether to go for a tool to check memory or to do the task manually.

Incase of doing manually, he will have to note down the current free resource available in the system, run the code or product and note the
memory consumption during the process, and the resource freed after terminating the process. This could be done using a table or any other matrix and an example is shown below:

<table>
<thead>
<tr>
<th>Before launching</th>
<th>After launching</th>
<th>During Execution</th>
<th>After Execution</th>
<th>After terminating</th>
</tr>
</thead>
<tbody>
<tr>
<td>93% free</td>
<td>91% free</td>
<td>85% free</td>
<td>87% free</td>
<td>89% free</td>
</tr>
</tbody>
</table>

From the above table we can see that there is a total loss of 4% of memory from the whole process. Information is also present as to how much memory is being consumed at each level of the process.

Using third party tools the task is reduced to a much greater level and information is given at a bit more detail as to which all module is called, how much memory is each module taking at the execution level, etc. Code Test from Applied Microsystems, Bounds Checker from Compuware, are some of the tools to name.

Load Testing:
Imagine you are testing a Web Site and need to confirm that the site does not crash even when 10,000 users, login to the site at the same time. “HOW IN THE WORLD AM I SUPPOSED TO DO THAT?” would be the first question that comes out of your mind. But whatever said and done, it is a part of the job and has to be accomplished.

Applications like Load Runner from Mercury, Rational Test Studio from Rational can help us achieve this goal. These applications simulate a real time scenario where in thousands of users login to sites or use products simultaneously, causing the page or application enormous load. And incase any error is thrown up or system crashes, a log is created with information specifying the same and factors that caused the problem to occur.
Basic introduction to actual testing and its implications:

Testing as such when considered in a broad band falls under two basic categories, namely, Product Testing and Installation Testing. It basically means that the software is tested in two broad categories mentioned above. It is on these two categories that different forms of testing is carried out, i.e., Black-box testing, Automation testing etc.

Let us look into these separately and in depth:

Product Testing:

As the name itself mentions, this category deals with the testing of the complete product inside out. The product is tested in different stages and in different methods.

During the initial testing of a new product, the Products are given to the QA department as modules. Modules will basically be the major functionality that the Product has to perform. Consider an example where in a Package comprises of more than one Product. In this case, the Modules will be either a Product or one of the major functionalities that one of the Products has to do.

The Test Cycle for Modules or Product is based on the features that need to be tested and the number of resources of the team that is looking into it. When a Package consists of more than one Product it is natural that the Testing team be divided into multiple teams each looking into a particular Product. Each of the team will be lead by a Team lead who co-ordinates with the Manager or Developer throughout the life of the Product.

When testing a Product, the features of the Product are verified thoroughly and confirmed that it is as per the specifications made by the Client or as per the Requirement spec that was given at the time of developing the Product. To accomplish this, Test Cases are created and run. Issues
noticed are reported to the developers and to the concerned higher officials and they are kept track of for further verifications.

In the Next cycle of testing, fixes made for the issues faced in the previous cycle are verified and also the complete product is verified again to make sure that what was working earlier is still working.

After a couple of cycles, a DRM (Defect Resolution Meeting) is conducted to reach to a conclusion on certain defects that have not been fixed.

In-between this, the modules are integrated and the complete product is developed. Now all the modules have to be rechecked to make sure that issues fixed are still in fixed state and those features those were working properly is still working properly. This check is known as Regression Testing. When modules are integrated, there will be a communication happening between these modules. The output of one of the modules could be the input of the other modules. So here the testing is done with the actual input and not with a dummy input. This leads to testing the integrated modules a bit more thoroughly. This is known as the Integrated Testing.

A Build (Installation media) might be made at this level and released to the QA department for testing. Once this is done then onwards when a major number of fixes have been made, then instead of receiving fixes in the form of files, files will be given only for fix verification and then another Build is made and released for testing.

This cycle is repeated until the Product reaches a Stable state where in it can be released to the market. The QA Manager, Product Manager, Development Leads and QA Leads decide the stability of the product.

Once a release has been made to the market, then the Product development does not stop there. Requirements can come in from the Clients and Issues that were not fixed for this release will be looked into.
This results again in a Development cycle of the product, which means that QA cycle is also to begin.

Based on the requirements from the Clients and Issues that are going to be fixed for the next release, QA starts updating their present Checklists and Test Cases or starts creating new ones before the first Deliverable from the Development team. And then the Test Cycle begins till the next release.

Now this testing has to be performed on all the OS (Operating Systems) that the company proposes to support because some of the functions used in creating the Product might not be available or might not behave the expected way in the other OS. It has been noticed that in case the developer works on Windows NT, then there could be some issues that are found only on other OS apart from Windows NT. Moreover, there could be some dependency files required for the Product to work that could have been missed out from the developer.

It is mostly in this category that the different kinds of testing is done, like Automation Testing, White-Box testing, Code Coverage, Memory Testing etc.

Installation Testing:

Installation testing is the process where in the Installation media is tested. Here it is tested to make sure that product gets installed on all Operating Systems with different configurations properly and that the correct Product files have gotten into the build. Installation testing is also called as Setup Testing.

This phase of testing is very crucial for the product because if the installation is not proper, then the Products installed will not function properly at all. This is not all there is to Installation Testing.

As mentioned, Installation testing confirms the installation of the Package on different OS (Operating systems), with different configurations. So what
exactly does this mean? Well, it means that the Package is installed using the media on the various platforms that it has been developed for, like Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Linux, etc. We shall consider only the Windows family in this process.

The Installation media could be made as a single media for all the Windows OS or separate media for the different Windows OS families. Windows OS families are categorized as Windows 9x family (Windows 95, 98 and ME), Windows 2000 and Windows XP.

Incase a single media is made for all the OS, then it is upto the Installation Testing team to confirm that it gets installed on all OS properly and works fine. Incase separate media are made for different OS, then the Installation team has to confirm that the media installs only on it's corresponding OS and not on the other OS.

This looks like a simple test. Is that all there is to Installation testing? Actually this does not cover the basics of it. Like mentioned earlier, different configurations of the machine is also looked into.

Usually when products are developed, a lot of it's features might require certain external files to be present in the machine. For ex: a Product using XML technology requires to have a minimum of IE5 on the machine as XML support comes only from IE5 and above. Certain other Packages might require that on NT Service Pack 4 and above be installed. Now all these checks have to be done by the Installation media, and handled properly. Also the minimum amount of space present in the machine etc are also looked into and handled properly.

Some Packages might install the product and the product might say that about the minimum requirements (which is hardly done nowadays), or else at the time of installation, the media itself detects that minimum requirements are not met and might abort setup. This required that the Installation media be tested with the minimum requirements and without
the minimum requirements. This deals with what is known as Fresh Installation of the Package.

Apart from minimum configuration, other configurations are also tested i.e. with IE5, IE5.5, IE6 etc, with Service Pack 4, 5, 6 etc and other versions of the minimum requirements. Also testing is done on a freshly formatted machine (which is called a Vanilla machine) and on used machines.

Apart from this, what if this is not the first Package that is being released. In this case, the installation team needs to make sure that upgrading the previous installation to the latest is possible and that this does not cause the Products to not work. All the latest files should have gotten onto the machine on doing an Upgrade, and all of the User settings that were made in the old Product version should still remain.

Installation testing is not just testing whether the Package is getting installed properly, but also to confirm that on Uninstalling, the Package references are removed completely from the machine. Note that here it means that only the installed files needs to be removed and not user created files. This has to be taken care at the point of uninstallation.

In short, uninstalling a Package should either ask the user whether the files he has created needs to be removed, else it should only remove files installed by the media and the registry entries that was created.

Other than Installation test and Uninstallation testing, the Installation Testing team has to confirm that the Products installed in the package are working fine and not causing any problems. This team does not do an in-depth testing. They just verify that all the basic functionalities of the products are fine.

Note that this testing has to be covered on all OS.
Testing: – Product Company vs. Project Company:

A Project company survives on the number of contacts that the company has and the number of Projects that the company gets from other forms. Whereas a Product based company’s existence depends entirely on how it’s product does in the market.

A Project Company will have the specifications made from the customer as to how the Application should be. Since a Project company will be doing the same kind of Project for some other Companies, they get to be better and know what are the issues and can handle them.

A Product company needs to develop it’s own specification and make sure that they are generic. Also it has to be made sure that the Application is compatible with other Applications. In a Product company, the application created will always be new in some way or the other, causing the application to be more vulnerable in terms of bugs. When upgrades are made for the different functionalities, care has to be taken that it will not cause any other module to not function. When any functionality is being implemented, have to make sure that they are introduced in such a way that it will not cause problems in the future for upgradation and for implementation of any other new functions.
Basic Skills required by a Tester:

Need to have a good Observation Skill. (Eye for details)
Need to have good Communication Skill.
People handling skills also required to some level. (To avoid conflicts with the Developers)
Good grasping power.
Patience.
Creativity in-terms of identifying problematic areas. (Which needs to be developed over the time of work)

Case Study of problems caused incase Testing not done properly: (All Case Study information from World Wide Web)

1. News reports in September of 2000 told of a software vendor settling a lawsuit with a large mortgage lender; the vendor had reportedly delivered an online mortgage processing system that did not meet specifications, was delivered late, and didn’t work.

2. In early 2000, major problems were reported with a new computer system in a large suburban U.S. public school district with 100,000+ students; problems included 10,000 erroneous report cards and students left stranded by failed class registration systems; the district’s CIO was fired. The school district decided to reinstate it’s original 25–year old system for at least a year until the bugs were worked out of the new system by the software vendors.

3. In October of 1999 the $125 million NASA Mars Climate Orbiter spacecraft was believed to be lost in space due to a simple data conversion error. It was determined that spacecraft software used certain data in English units that should have been in metric units. Among other tasks, the orbiter was to serve as a communications relay for the Mars Polar Lander mission, which failed for unknown reasons in December 1999. Several investigating panels were convened to determine the process failures that allowed the error to go undetected.
4. A small town in Illinois received an unusually large monthly electric bill of $7 million in March of 1999. This was about 700 times larger than it’s normal bill. It turned out to be due to bugs in new software that had been purchased by the local Power Company to deal with Y2K software issues.

5. January 1998 news reports told of software problems at a major U.S. telecommunications company that resulted in no charges for long distance calls for a month for 400,000 customers. The problem went undetected until customers called up with questions about their bills.

6. A retail store chain filed suit in August of 1997 against a transaction processing system vendor (not a credit card company) due to the software's inability to handle credit cards with year 2000 expiration dates.

7. Software bugs in a Soviet early-warning monitoring system nearly brought on nuclear war in 1983, according to news reports in early 1999. The software was supposed to filter out false missile detections caused by Soviet satellites picking up sunlight reflections off cloud-tops, but failed to do so. Disaster was averted when a Soviet commander, based on what he said was a ‘...funny feeling in my gut’, decided the apparent missile attack was a false alarm. The filtering software code was rewritten.

8. WASHINGTON -- Software bugs are costing the U.S. economy an estimated $59.5 billion each year, with more than half of the cost borne by end users and the remainder by developers and vendors, according to a new federal study. (Source – Computer World)