

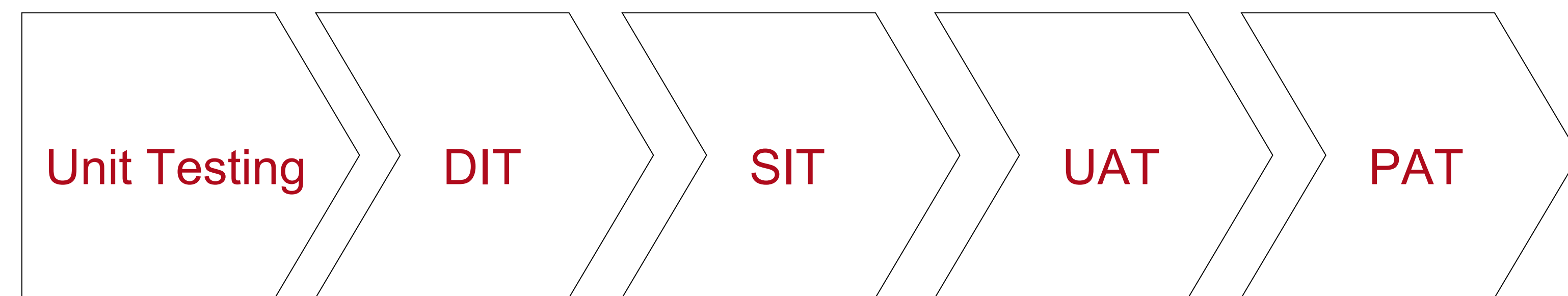
## Learning Objectives

- Be able to use the Mercury Quality Center package effectively in order to verify that an application works as required
- To become more proficient at writing concise test cases and documenting defects thoroughly
- Apply information gathered from the current defects in order to provide possible causes of these issues

## Effective Testing

- *test to prove functionality* – try with perfect data to make sure it works as expected
- *test to break the system* – try with invalid data (i.e. type text where numbers are supposed to be)
- *test based on the areas of greatest risk* – things with greater risk of failing should be tested as exhaustively as possible

## Stages of Testing

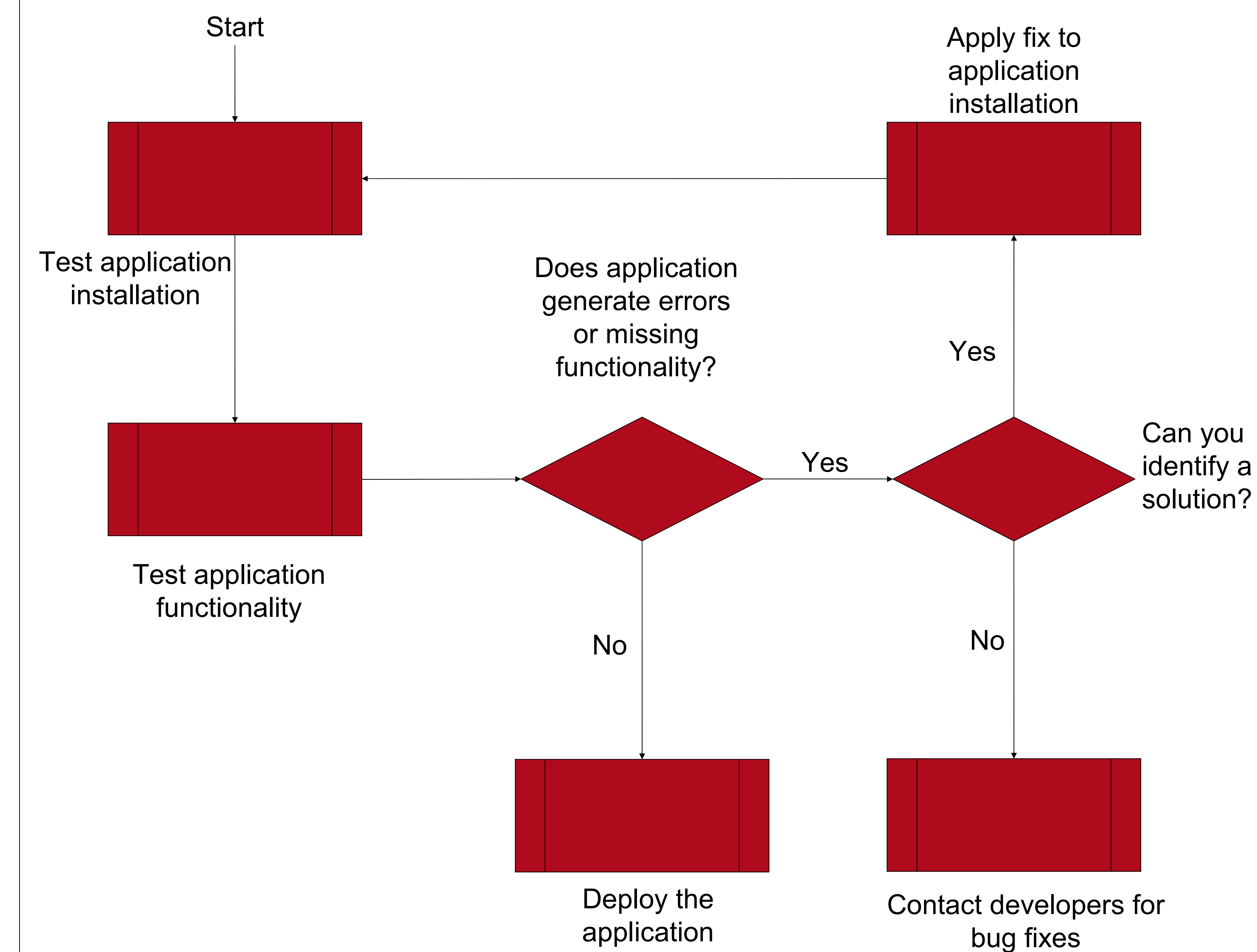


Test Stage	Completed By	Tests
Unit Testing	• Developers	• Code (one component/part)
DIT (Developer Integration Testing)	• Developers	• Connectivity of the parts of code.
SIT (Systems Integration Testing)	• Test Analyst Team	• Total end-to-end functionality • Attempt to break the system • Business Requirements as mentioned in requirements document
UAT (User Acceptance Testing)	• Test Analyst Team • Business Team	• Total end-to-end functionality • Business Requirements as mentioned in requirements document • Confirm functionality of Day-to-Day tasks
PAT (Production Acceptance Testing)	• "PAT" Team	• Operability • Availability • Security • Performance

## Testers – Roles and Responsibilities

- Normally involved at the beginning of the SIT/DIT testing stages
- Once project requirements have been determined, one or two testers are assigned the task of creating test cases (based on the requirements document) which are used to test the application as much as possible
- Testers are present in order to execute the test cases as well as log any defects they may have found throughout the time running these test cases
- Designated testers help in the PAT process in order to confirm that the installation went as planned

## Testing Process



## Test Case Dos and Don'ts

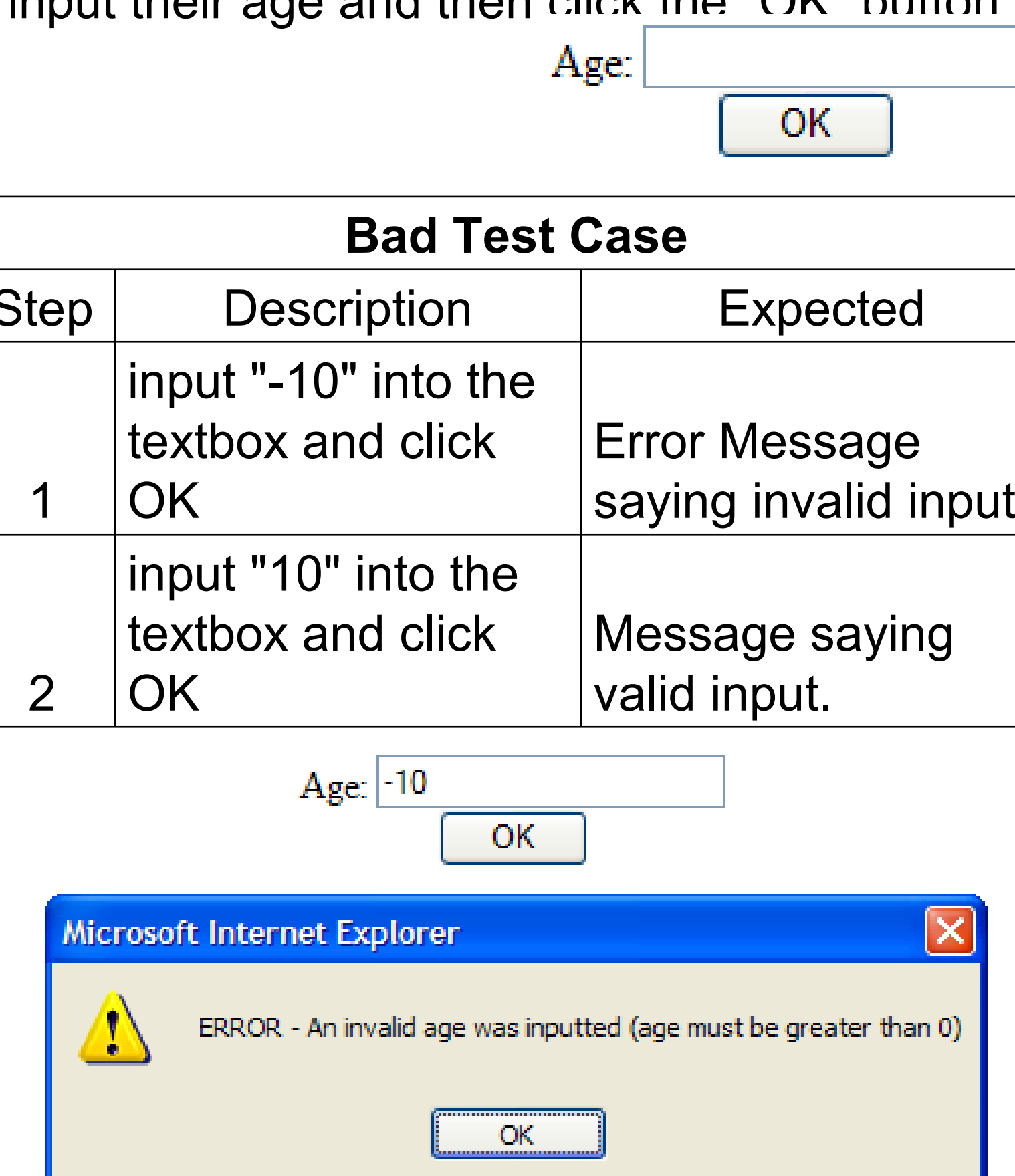
Do	Don't
✓ test with perfect data*	✗ test solely with perfect data*
✓ try to break the system (i.e. use invalid data)	✗ assume that if it worked properly once, it will always work perfectly
✓ regression test (ensure previous parts of the application still work after new parts have been added)	
✓ try to make reusable parts of test cases	

\*perfect data – data which is exactly what the application is looking for

## Good Testing vs. Bad Testing

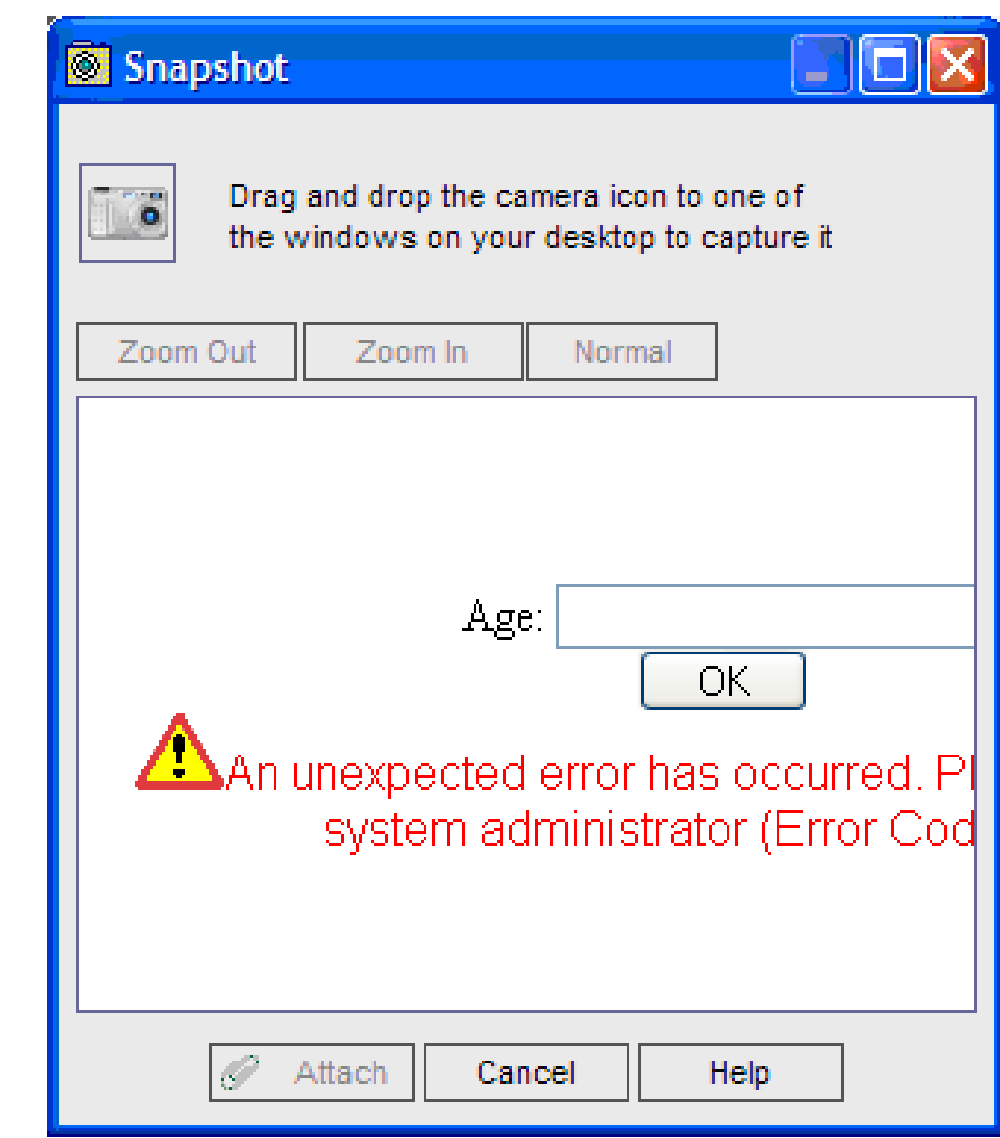
**Scenario** – you have a text box where the user is to input their age and then click the "OK" button

Good Test Case			Bad Test Case		
Step	Description	Expected	Step	Description	Expected
1	input "abc" into the textbox and click OK	Error Message saying invalid input.	1	input "-10" into the textbox and click OK	Error Message saying invalid input.
2	input "a2c" into the textbox and click OK	Error Message saying invalid input.	2	input "10" into the textbox and click OK	Message saying valid input.
3	input "-10" into the textbox and click OK	Error Message saying invalid input.			
4	input "#@?" into the textbox and click OK	Error Message saying invalid input.			
5	input "10" into the textbox and click OK	Message saying valid input.			



## Screenshots in Testing

- Help point developers in the right direction when they are trying to debug defects
- Prove that the application was working properly at a particular stage of the project
- Prove you are doing your job properly



## Mercury Quality Center

- The testing tool which was used throughout my work term at CIBC.
- It can be used to store the following:
  - *requirements* – useful for linking each specific test case to a requirement
  - *test cases* – allows the testers to make and run test cases
  - *defects* – allows you to keep track of any issues which arise from the execution of test cases (status to keep track of what is happening with any given defect)



## Key Terms

- *Testing* – a method of verification used to ensure functionality of an application
- *Bug* – An error in computer programming, for example incorrect coding of an instruction (syntax error) or instructions unable to provide the required solution to a particular problem (logic error)
- *Requirements Document* – a document which is written by the business for the developers in order to say what they want out of the project. This document is what testers use to get a general sense of direction of what they will be testing as well as what their test cases should encompass
- *Test Case* – a set of test steps which are used in the testing of an application, written by the tester
- *Testing Tools* – these are tools which are used in order to make a tester's life easier

## References

- <http://www.eubios.info/biodict.htm#b>
- [http://www.microsoft.com/china/technet/images/itsolutions/cits/dsd/bdd/images/bddsup07\\_BI\\_G.gif](http://www.microsoft.com/china/technet/images/itsolutions/cits/dsd/bdd/images/bddsup07_BI_G.gif) (Modified version of their testing process)
- [http://en.wikipedia.org/wiki/Software\\_testing](http://en.wikipedia.org/wiki/Software_testing)
- <http://www.onestoptesting.com/introduction/life-cycle.asp>
- [http://www.ece.cmu.edu/~koopman/des\\_s99/sw\\_testing/](http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/)

## About this Work Term

- First work term
- Halfway through second year of B. Sc. Computer Science
- Worked at Canadian Imperial Bank of Commerce as a Test Analyst, Toronto, Ontario
- Tested the Online Business Banking section of CIBC