DEBUGGING&TRACEABILITY

Date	12 November 2022
Team Id	PNT2022TMID43265
Project Name	Efficient water quality analysis & predicition using machine learning
Mark	2

INTRODUCTION

Comprise of two words i.e. trace and ability. Trace means to find someone or something and ability means to a skill or capability or talent to do something. Therefore, traceability simply means the ability to trace the requirement, to provide better quality, to find any

Traceability

. Due to this, it's easy for the suppliers to reduce any risk or any issue if found and to improve the quality of the item or product. So, it's important to have traceability rather than no traceability.

Source traceability

These are basically the links from requirement to stakeholders who propose these requirements traceability

These are the links between dependent requirements.

Design traceability

These are the links from require

Debugging, Tracing, and Profiling frequently executed Profiling evaluates and describes the lines of source code that generate the most, and how much time it takes to execute them.

. NET Framework applications are easily debugged by using Visual Studio, which handles many of the configuration details. If Visual Studio is not installed, you can examine and improve the performance of .NET Framework applications by using the debugging classes in the .NET Framework System. Name space. This namespace includes, and classes for tracing execution flow, and the Process and Perform classes for profiling code.

CONCLUSION

All kinds of techniques and tools allow engineers to root out problems within a software environment.

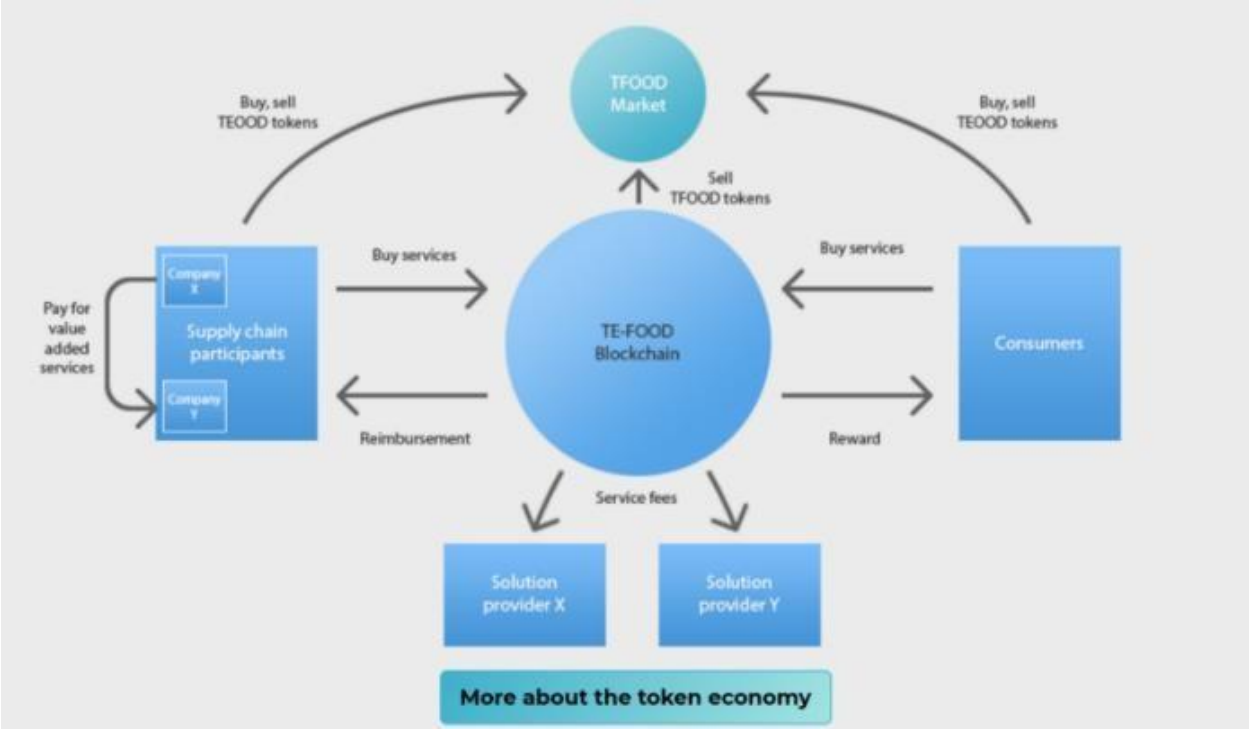
Conditional Debug Attributes

Conditional attributes, like other C# attributes, do not put any burden on the code calling them. For example, in Listing 21.8, if you put #define I at the top of your code or compile your code with the /D:I want DEBUG option, then Method can be called. Otherwise, the Microsoft intermediate language code for the my

Method code will be omitted (not generated) in your final assembly

C Development Guide

Provides a guide to all key technology areas and tasks for application development, including creating, configuring, debugging, securing, and deploying your application, and information about dynamic programming, interoperability, extensibility



Description of technique

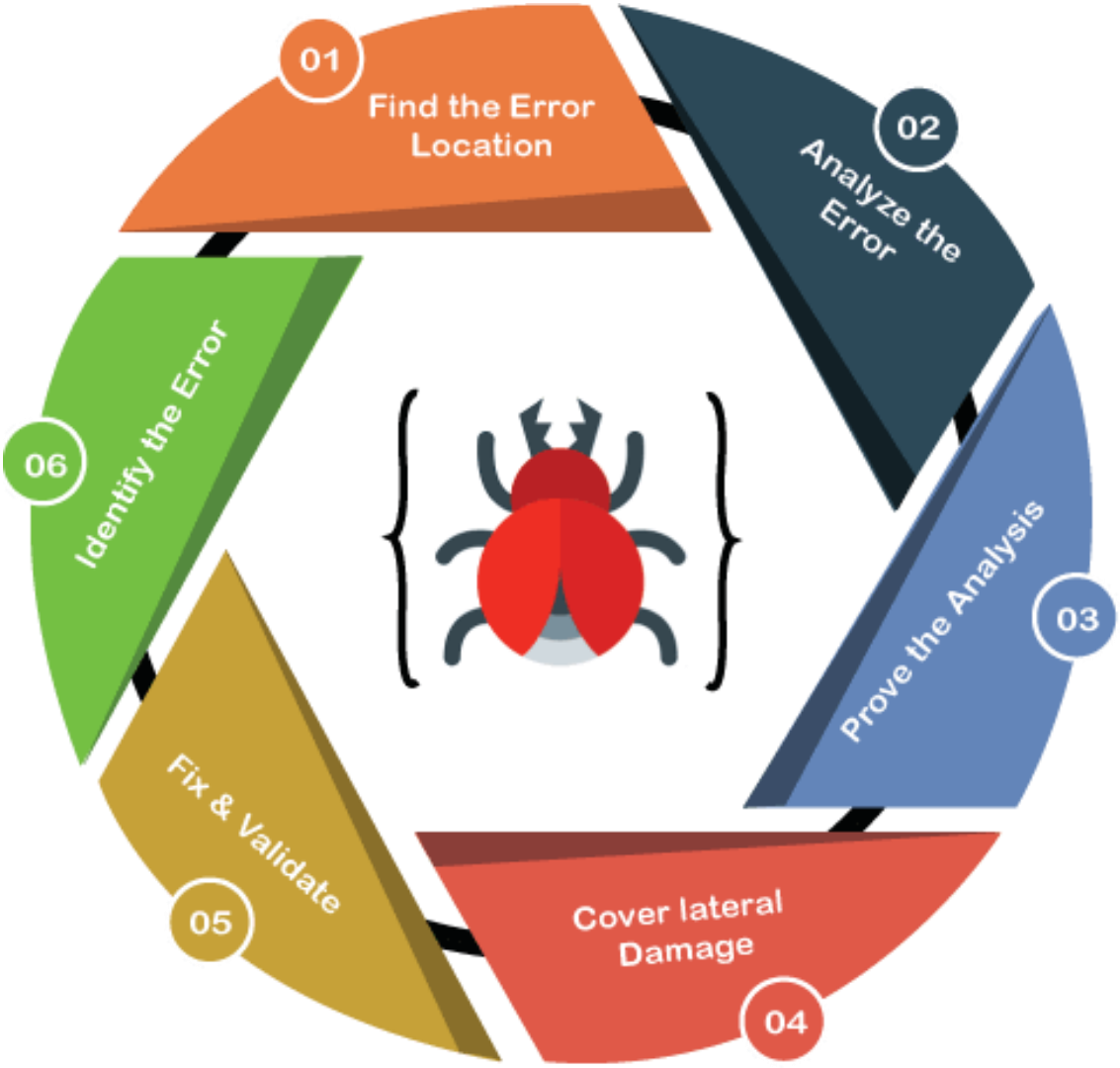
The steps in the Create a Sample with the Debug Class section demonstrate how to create a console application that uses the Debug class to provide information about the program execution.

When the program is run, you can use methods of the Debug class to produce messages that help you to monitor the program execution sequence, to detect malfunctions, or to provide performance measurement information. By default, the messages that the Debug class produces appear in the Output window of the Visual Studio Integrated Development Environment (IDE).

The sample code uses the Write Line method to produce a message that is followed by a line terminator. When you use this method to produce a message, each message appears on a separate line in the Output window.

When you use the Assert method of the Debug class, the Output window displays a message only if a specified condition evaluates to false. The message also appears in a modal dialog box to the user. The dialog box includes the message, the project name, and the Debug. Assert statement number. The dialog box also includes the following three command buttons:

Steps involved in Debugging



Identify the Error:

Identifying an error in a wrong may result in the wastage of time. It is very obvious that the production errors reported by users are hard to interpret, and sometimes the information we receive is misleading. Thus, it is mandatory to identify the actual error.

Find the Error Location:

Once the error is correctly discovered, you will be required to thoroughly review the code repeatedly to locate the position of the error. In general, this step focuses on finding the error rather than perceiving it.

Analyze the Error

The third step comprises error analysis, a bottom-up approach that starts from the location of the error followed by analyzing the code. This step makes it easier to comprehend the errors. Mainly error analysis has two significant goals, i.e., evaluation of errors all over again to find existing bugs and postulating the uncertainty of incoming collateral damage in a fix.

Debugging Strategies

- For a better understanding of a system, it is necessary to study the system in depth. It makes it easier for the debugger to fabricate distinct illustrations of such systems that are needed to be debugged.
- The backward analysis analyzes the program from the backward location where the failure message has occurred to determine the defect region. It is necessary to learn the area of defects to understand the reason for defects.
- In the forward analysis, the program tracks the problem in the forward direction by utilizing the breakpoints or print statements incurred at different points in the program. It emphasizes those regions where the wrong outputs are obtained.

- To check and fix similar kinds of problems, it is recommended to utilize past experiences. The success rate of this approach is directly proportional to the proficiency of the debugger.