Enhancement in Nunit Testing Framework by binding Logging with Unit testing

Viraj Daxini[1], D. A. Parikh[2]

[1]Gujarat Technological University, L. D. College of Engineering, Ahmedabad, Gujarat, India

[2]Associate Professor and Head, Department of Computer Engineering, L. D. College of Engineering, Ahmadabad, Gujarat, India

Abstract: ***This paper describes the new testing library named log4nunit testing library which is enhanced version of nunit testing library as it is binded to logging functionality. NUnit is similar to xunit testing family in all that cases are built directly to the code of the project. Log4NUnit testing library provides the capability to log the results of tests along with testing classes. It also provides the capability to log the messages from within the testing methods. It supplies extension of NUnit's Test class method in terms of logging the results. It has functionality to log the entity of test cases, based on the Logging framework, to document the test settings which is done at the time of testing, test case execution and it's results. Attractive feature of new library is that it will redirect the log message to console at runtime if Log4net library is not present in the class path. It will make easy for those users who do not want to download Logging framework and put it in their class path. Log4NUnit also include a Log4net logger methods and implements utility of logging methods such as info, debug, warn, fatal etc. This work was prompted due to lack of published results concerning the implementation of Log4NUnit.***

Keywords: XUnit, NUnit Testing, Log4NUnit, logging, log4net.

1. Introduction

Log4NUnit is an OpenFuture Project [1]. As described by Per Runeson [2] in their survey of Unit testing Practices they define unit testing and evaluate their strengths and weaknesses. They listed Definitions Strengths and Problems with the Unit testing and one of the problems they identify during their survey is Unit testing does not have proper documentation. The purpose of binding logging framework with unit testing is the necessity to create test protocols for Nunit[4]. Nunit testing requires test cases and test fixtures to identify the test are correct or not. The entire testing tool does not support any kind of logging functionality for successfully executed test cases. So to keep track of the testing processes and for monitoring the performance of test cases, successful execution of a test case, results of successful testing and bugs which are identified during the testing should be further documented. So the basic intention behind building Log4NUnit as shown in Figure 1 is the necessity to create test protocols in order to document the entire initial test setting, test case execution and results of successful test case. Along with the testing framework (NUnit testing, JUnit testing) a logging component (Log4J, Smart Inspect) is also required to document the results of test case executions. One of the greatest logging libraries are there for .net is Smart Inspect and Log4Net. In addition to the core feature available in log4net some advance features are also there like ADO.net, LINQ queries and .net Trace and debug API [3]. Programmers generally using some logging framework for logging their test case and results of successful testing instead they can use this library which provides logging facility along with the testing of ongoing project. Log4NUnit library provides capability of logging message along with the testing. By using the pre-generated test report programmers can easily do further testing and hence efficiency will be increase. It will also generate the performance matrix of successfully executed test case. Which can further been used for analysis purpose. The output of Log4Nunit testing library can be directly applied as an input to any of the configuration management tool like jeera.The rest of the paper is organized as follows. In section 2, the paper describes the background knowledge required to understand the Log4NUnit structure. Section 3 will demonstrate how to make use of it in different programs (for code and detailed steps please contact corresponding author of the paper). In Section 4, emphasis is given to the advantages for using this library & in section 5 various recommendations are outlined for future work on it. In the end section 6 illustrates the conclusions of this paper and section 7 consists of acknowledgments.

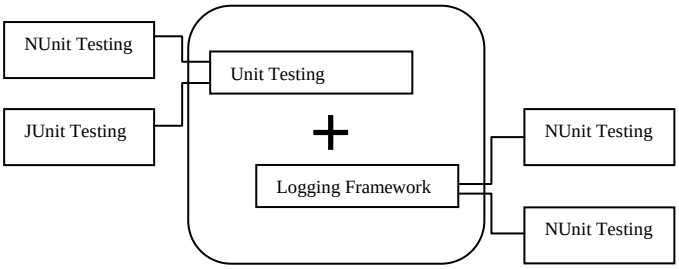


Figure 1: Log4Unit as an Integration of Nunit and Logging

2. Background Work

According to Sira vegas as describe in A Case study of Unit testing Techniques [5], Log4NUnit is solution for logging messages from within the test cases. It provides us with an extension of NUnit's Test class. At the time of production, developer cannot debug the complete application and it may be possible that some errors which appear in production server don't appear in development or in the preprod server. So if you try to solve the error, you need to find a proper way which will help you in identifying the error and tracing

your application at the time production without stopping or using debug mode in the application. Log4net is a readymade tool for the programmer to output log statements to variety of output which targets in production, preprod or to the development server. There are many ways For Installing and configuring the tool. This tool is able to work with ASP.NET 2.0, 3.0, 3.5 using C#.

Log4net will help you for collecting errors in following ways:

- Email Log Using SMTPAppender
- XML and XLS log Using FileAppender
- Using EventLogAppender to log particular event
- Using ADONetAppender to log the results in data base

There are several kinds of logging levels supported by log4net library which are Off (), Error (), Warn (), Fatal (), Debug (), Info () and All-everything gets logged.
Above logging methods are provided in the Log4net library for various log levels. Here the log method itself is not provided; instead each log level method needs to be used directly. It can be said that the design pattern known as Adapter [6] is used to design this class. It uses a concept where Object Adapter relies on object composition as shown in Figure 2. Setup and Teardown methods are also provided.

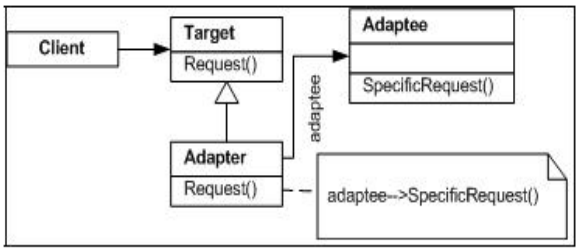


Figure 2: Object Adapter design pattern diagram from [6]

In NUnit, preparation for testing is called setup. It's independent of the assertions, so the same scaffolding can apply to several individual tests. The setup is run before each test. When a test ends, the scaffolding might require some cleanup actions.

3. Integration of NUnit with Logging and Results

Lots of people head the project into automation and they eventually start writing different scripts. I have experienced, that is a little rash as some infrastructure bits needs to be take care are present.
The first thing that we need to keep in mind according to moonzoo kim[7] is to identify the location of the scripts which will be stored in the version control. Everything needs to be there in version control for providing automation. (If anyone has proposed the solutions consultants who don't have the access to client content but write code.) This should be the first step.
The next step is to identify the logging strategy. The reason behind doing this is twofold [8]. First, it will give you means of debugging the script and second it will allow you to create the record for test execution, result and test failure details for latter reference.
The proposed algorithm for Binding logging and Unit testing is shown in Figure 3. You need to follow the steps given in the algorithm and write the script for the same in the source of Unit testing framework. (For the complete source code of Algorithm you can contact author of this paper).

Proposed Algorithm for Binding Logging to Unit Testing Framework

- Step-1 Capture Logging Information
- Step-2 Identify the logging levels
- Step-3 Identify the logging information and make decision that which information to be logged
- Step-4 Prepare Final logging information
- Step-5 Format logging information in different style
- Step-6 Publish logging information to various destination
- Step-7 End

Figure 3: Proposed Algorithm of Log4Nunit Library

Figure 4 shows the enhanced version of NUnit testing tool. This is having some functionality to log the message. By selecting the DropDownList user can able to generate log file in different format like HTML, CSV, XML, XLS and Text format. It also support the functionally to generate the performance matrix for successfully executed test case. By Clicking on a Button Performance Matrix it will generate Graph as Shown in Figure 6. A second approach is there, In case if user does not want to write the complete script, they can modify the source by adding logging framework. There are verity of logging framework available for .net project but among those most widely used framework for logging is log4net and an application need to configure it in order for your instructions, such as where to log the message, what to ignore with it, etc, are to be carried out. However, there are several characteristics that one has to be aware about, particularly when we are using it with NUnit testing tool. Currently i am using NUnit version 2.5.7 for testing framework and log4net version 1.2.10 as a logging framework and hence the configuration below is relevant to the combination of this version only.

A Problem i have identify when using in Nunit, a report has been failed and it cannot generate any log file. In other word, we can say that the combination fails to find the required configuration file. For specifying the configuration information, an assembly can able to include any custom assembly-level attribute like Log4Net, XmlConfiguratorAttribute. For the situation where we need to load the configuration file, the internal logic of log4net will look for an instance of this custom attribute in a given assembly for the configuration file. This Configuration file is located in path src/Config/XmlConfiguratorAttribute.cs.

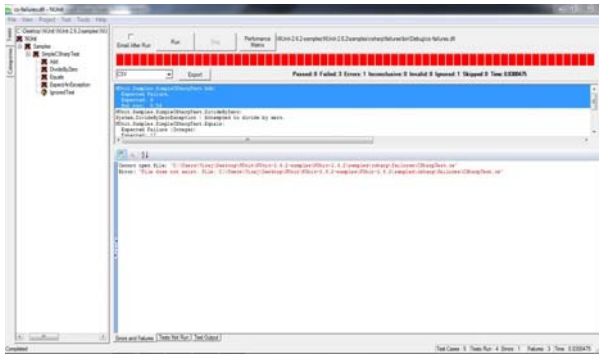


Figure 4: Enhanced NUnit testing tool with logging

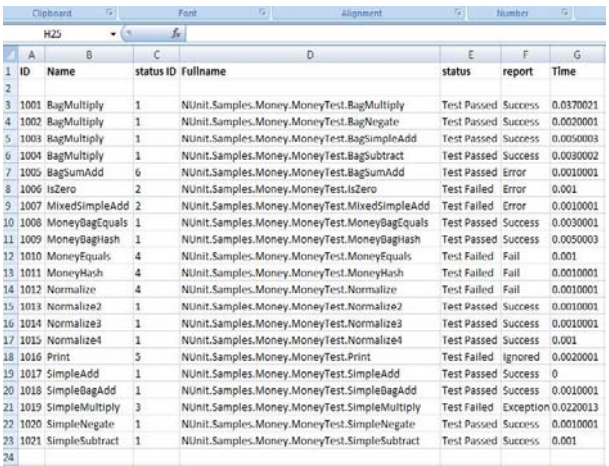
To use Log4NUnit you need to follow some guidelines which are given.

- Include the assembly attribute for log4net
- Add the log4net reference to your visual studio project
- Add Log4net bits at the script
- Configure the log4net tool by adding new config. file to your projectlog4net.config

If you had done the entire thing mentioned about correctly, log4net will never find this file because it is expected by default in application configuration or web configuration file. So we need to explicitly add the log4net where its configuration file can be found for that we need add the assembly given below to our Assembly info.cs file.

After doing this you have successfully integrated log4net library to your project and you can easily send the log messages to it by using different methods like log.Debug(), log.Info() and log.Error(). NUnit testing framework will be the default capture for logging messages as pass they along to the user if they are Error or higher, regardless of what log4net tells. So in that particular case we need one another Application Configuration File to be added. But if you run the NUnit scripts from the GUI of NUnit you can easily specify the configuration file which you want.

If you had done all above mentioned things correctly with integrating the Log4net and Nunit testing you have created the Log4NUnit Library which can able to give the logging facility like logging Initial test setting, test results and error



	A	B	C	D	E	F	G
1	ID	Name	status ID	Fullname	status	report	Time
2							
3	1001	BagMultiply	1	NUnit.Samples.Money.MoneyTest.BagMultiply	Test Passed	Success	0.0370021
4	1002	BagMultiply	1	NUnit.Samples.Money.MoneyTest.BagNegate	Test Passed	Success	0.0020001
5	1003	BagMultiply	1	NUnit.Samples.Money.MoneyTest.BagSimpleAdd	Test Passed	Success	0.0050003
6	1004	BagMultiply	1	NUnit.Samples.Money.MoneyTest.BagSubtract	Test Passed	Success	0.0030002
7	1005	BagSumAdd	6	NUnit.Samples.Money.MoneyTest.BagSumAdd	Test Passed	Error	0.0010001
8	1006	IsZero	2	NUnit.Samples.Money.MoneyTest.IsZero	Test Failed	Error	0.001
9	1007	MixedSimpleAdd	2	NUnit.Samples.Money.MoneyTest.MixedSimpleAdd	Test Failed	Error	0.0010001
10	1008	MoneyBagEquals	1	NUnit.Samples.Money.MoneyTest.MoneyBagEquals	Test Passed	Success	0.0030001
11	1009	MoneyBagHash	1	NUnit.Samples.Money.MoneyTest.MoneyBagHash	Test Passed	Success	0.0050003
12	1010	MoneyEquals	4	NUnit.Samples.Money.MoneyTest.MoneyEquals	Test Failed	Fail	0.001
13	1011	MoneyHash	4	NUnit.Samples.Money.MoneyTest.MoneyHash	Test Failed	Fail	0.0010001
14	1012	Normalize	4	NUnit.Samples.Money.MoneyTest.Normalize	Test Failed	Fail	0.0010001
15	1013	Normalize2	1	NUnit.Samples.Money.MoneyTest.Normalize2	Test Passed	Success	0.0010001
16	1014	Normalize3	1	NUnit.Samples.Money.MoneyTest.Normalize3	Test Passed	Success	0.0010001
17	1015	Normalize4	1	NUnit.Samples.Money.MoneyTest.Normalize4	Test Passed	Success	0.001
18	1016	Print	5	NUnit.Samples.Money.MoneyTest.Print	Test Failed	Ignored	0.0020001
19	1017	SimpleAdd	1	NUnit.Samples.Money.MoneyTest.SimpleAdd	Test Passed	Success	0
20	1018	SimpleBagAdd	1	NUnit.Samples.Money.MoneyTest.SimpleBagAdd	Test Passed	Success	0.0010001
21	1019	SimpleMultiply	3	NUnit.Samples.Money.MoneyTest.SimpleMultiply	Test Failed	Exception	0.0220013
22	1020	SimpleNegate	1	NUnit.Samples.Money.MoneyTest.SimpleNegate	Test Passed	Success	0.0010001
23	1021	SimpleSubtract	1	NUnit.Samples.Money.MoneyTest.SimpleSubtract	Test Passed	Success	0.001
24							

Figure 5: Log File Generated by Log4Nunit Library

-result can be logged in a text format, CSV format, XML Format, XLS format and HTML format. It will also Generate Performance Matrix of Successfully executed test cases. Figure 5 Shows logfile in XLS format and Figure 6 show performance Matrix of Test pass/ Fail.

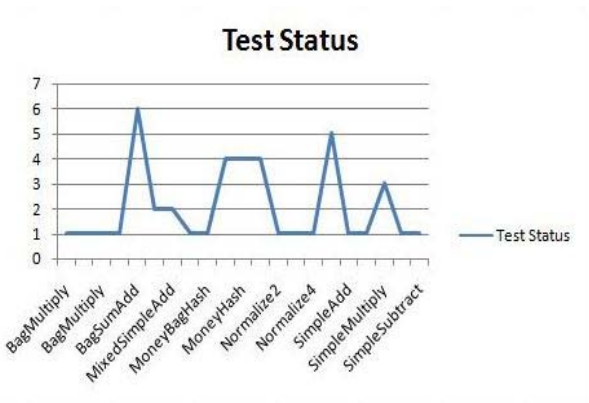


Figure 6: Performance Matrix Generated by Log4Nunit Library

4. Benefits of using Log4Nunit Library

1 By Using Log4Nunit testing library Software tester can log unit testing module at several log levels.
2 Log4Nunit testing library Can able to record the logging event in common log file with different format like it can generate log file in XML format, text format, CSV Format and many more.
3 As log4Nunit testing library generates report in terms of log file of successfully executed test case it may be useful for identifying why test fail.
4 Lof4NUnit testing library provide attractive feature of generating performance metrics which can further be used for analysis purpose.
5 Log4Nunit testing library generates report for Test passed of Failed, so it can be useful to software tester when creating new test case or modifying the old one.
6 Log files log files generate by the unit testing can further be applied to configuration management tool (JIRA, Testlink etc.) to manage and track your software testing efforts.
7 It can also be useful for logging result of large test suites.

5. Recommendations for Future Work

1 Log4Unit is an open source and small library, so developers can customize it to support the logging implementation of their choice i.e. If any programmer wants to use any other logging framework, for example take SmartInspect [9], than they can change the Log4NUnit source and create their own library providing customized logging backend.
2 Another characteristic of Log4NUnit is that should be looked into is that, it becomes tightly coupled with the underlying logging framework. To overcome these inflexible situation programmers can use a logging framework like ObjectGay Framework [10]. ObjectGay will serve as a simple logging framework or abstraction for various logging

framework, which allows the user to plug in the desired logging framework at the time of deployment.

3 Scarcity of documentation on Log4NUnit makes it difficult for programmers and research students to understand its structure and take a look at the Software engineering concepts used in its structure design. So a lot of work needs to be done for creating the class diagrams and other documentation on this topic so that it can be used for further research.

6. Conclusion

This paper presented the enhancement in Unit testing framework by binding logging with Unit testing will generate a new library whose focus is to create test protocols for NUnit testing framework. It is being observed that the library is asymmetrical in the sense that it focuses on the documentation of test failures and errors. It is also tightly coupled with the underlying logging framework log4net. This library is very much useful in small scale projects that need to startup immediately with a Testing mechanism together with a logging framework. It also reduces the efforts of programmer by maintaining a separate logging framework. By using this library one can reduces time dependency as there is no need to generate separate testing report of test case, results and failure. This library is already been used by some developers and had proved useful to small scale projects. Since the Library is open source and it can also be customized to support the business requirements of individual organizations. Programmers can abstract its logging implementation by using a SmartInject or Nlog so as to decouple it from the logging library, thus making it independent of underlying logging framework.

References

[1] Log4Unit, “Open Future Project”. Available: http://www.olex.openlogic.com

[2] Per Runeson, “A survey of Unit testing Practices” By the IEEE computer Society, pp. 22-29, 2006.

[3] Bill Hamilton, NUnit Pocket Reference, Up and running with Nunit, O'reilly, 2004.

[4] NUnit, “Unit Testing for .net projects,” [Online]. Available: http://www.nunit.org (NUnit version 2.5.7)

[5] Log4net, “Leg4Net Logging tool,” [Online] Available: http://logging.apache.org/log4net/download (Log4net version 1.2.10)

[6] Sira vegas, Natalia Juristo and Vector R. Basili, “A case study on Unit Testing Techniques” IEEE Transactions on Software Engineering, VOL. 35, NO. 4, pp. 551-565, July 2009.

[7] Gamma, Erich; Richard Helm; Ralph Johnson; and John Vlissides, Design Patterns. Addison-Wesley, 1995.

[8] Moonzoo kim, Yunho kim and Hotae kim,, " A Comparative study of Software model checkers as a unit testing tools," An Industrial case study. IEEE Transactions on Software Engineering, VOL. 37, NO. 2, pp. 146-160, March 2011.

[9] SmartInspect, “Advanced High-Performance Logging and Tracing for .NET” [Online]. Available : http://www.gurock.com/smartinspect

[10] Objectgay, “.NET logging framework” [Online]. Available: http://www.theobjectguy.com/dotnetlog.

Author Profile

Viraj Daxini received the B.E. (CE) degree in 2009. He was working as a lecturer in from 2009-2011. He is now pursuing M.E. (IT) from L.D. College of Engineering, Ahmedabad. He is doing research in Software Testing and has published 2 International papers

.

Prof. D.A. Parikh is HOD of Computer Engineering Department in L.D. College of Engineering, Ahmadabad and has more the 22 years of teaching experience. He has guided so many post graduate students during his academic carrier. He has published more the 15 National and International Research papers.