Individual Report
Unit Test Visualisation Project

Author: Keagan Phillips
Team: Group 4
Course: ELEN7046 - Software Technologies and Techniques
Date Submitted: 25 June 2012
Source & Documentation:
https://github.com/KeaganPhillips/Wit-Group-4-project

Abstract

This document discusses the technologies and techniques used in the development of the "Unit Test Visualisation" project. It briefly discusses the specification and proposed solution. It also highlights the individual contribution made by the author.

Contents

1 Introduction

As per the ELEN 7046 course, students were required to form groups and develop a system that can be used for software visualization and/or education. Teams were given the freedom to formulate their own specification.

1.1 Specification selection

After much consideration the team agreed to develop a system that can visually represent unit tests to aid developers. Unit tests ensure that a system behaves in a consistent and predictable manner especially as new features are added or existing features are amended. The team however felt that the use of unit tests can be expanded to also describe software.

1.2 Proposed Solution

The proposed solution was for the development of a system that can visually render class diagrams on within a web browser. Upon clicking a class diagram, the developer will get the full context of each test for that class without having to zoom into each test within the code base one at a time.

2 Role and Responsibilities

I assumed the lead developer role within the team. As lead developer, I had three core goals i.e. quality, mentorship and delivery.

2.1 Quality

I had to ensure that all developers conform to good coding patterns and practices by overseeing the work being done by other software engineers on the project.

2.2 Mentorship

I also acted as a mentor to guide developers towards using best engineering practices while still being open to new ideas and suggestions as not to stifle creativity within the team.

2.3 Delivery

Ultimately the responsibility for delivering the end product rested on my shoulders. To ensure that the system is technically sound, meet the expected requirement and is delivered on time.

3 Individual Contribution

As developer lead I had oversight throughout the code base as a whole. I did however try not to dictate to the other developers by allowing them to be creative and to contribute meaningfully. The 'Unit Test Reflector' component was one component I worked on in isolation. The following sections briefly describe some of the core components within the system and my contribution to them.

3.1 Core System Under Test

This is a Microsoft Dot Net Framework 4.0[1] console application. This application forms the core business system (i.e. The system under test). The Unit Test project is intended to target this application in order to verify it's correctness. The team did not focus to much on getting this application fully functional because it falls beyond the scope of the project. The project focuses on the visualisation of unit tests, not the functionality of the core system under test.

This application was written in using the TDD (Test Driven Development[2]) technique. TDD was a bit difficult to grasp for some of the team members at first, but team really got used to the technique as the project progressed.

The group decided on a demo bank application as an illustration. Again the functionality and completeness of this project is beyond the scope of "Unit Test Visualisation". Our unit tests must have a system to test. This console bank application is therefore the test subject.

All the developers on the team collectively contributed to this project. My role was to provide mentorship and to guide the other developers with respect to TDD. Because TDD was new for most developers in the group, the team decided to employ the pair programming[3] technique for the development of this project. This technique worked very well as it aided a lot in terms of rapid knowledge transfer with respect to TDD. The developers (all four)

would sit together and one would write the code the TDD way. We would then rotate the developer doing the coding every few moments.

3.2 Core System Unit Tests

This compiles to a Microsoft dot Net framework 4.0[1] .dll file. It is the test project containing automated unit tests targeting the "Core System Under Test". Test must follow the unit test coding convention[4] in order for the framework to function correctly.

TDD is a process whereby the developer first write an automated test before writing the actual application code. All developers on the team participated in the development of this component as the team follow the TDD process. Because of the nature of TDD, the 'Core System Under Test' component and this component [Core System Unit Tests] was worked on concurrently by all developers in the team.

Again, as lead developer I played the role of mentor to guide the development team towards implementing TDD.

3.3 Unit Test Reflector

This component is responsible for extracting the automated unit test metadata from the "Core System Unit Tests" component. This is achieved by exploiting the "Reflection[5]" feature of the Microsoft dot Net framework 4.0[1].

C# Reflection is technique used to parse the metadata (e.g. methods, classes, types) of a compiled dll file. This is very powerful as it allows one to write code capable of reading another cosebase by inspecting the internal elements of that compiled assembly.

The task to develop this component was assigned to me. This component builds a C# data structure using the metadata extracted from the unit tests. Figure 1 shows a class diagram of the data structure it creates.

As depicted in the class diagram (Figure 1), this component creates a data C# structure that consists of an array of 'ClassUnderTest' objects. Each 'ClassUnderTest' object represents a class for which we have written at least one unit test. The 'ClassUnderTest' object contains fields (i.e. public methods / public properties / class name) used to render the class diagrams on

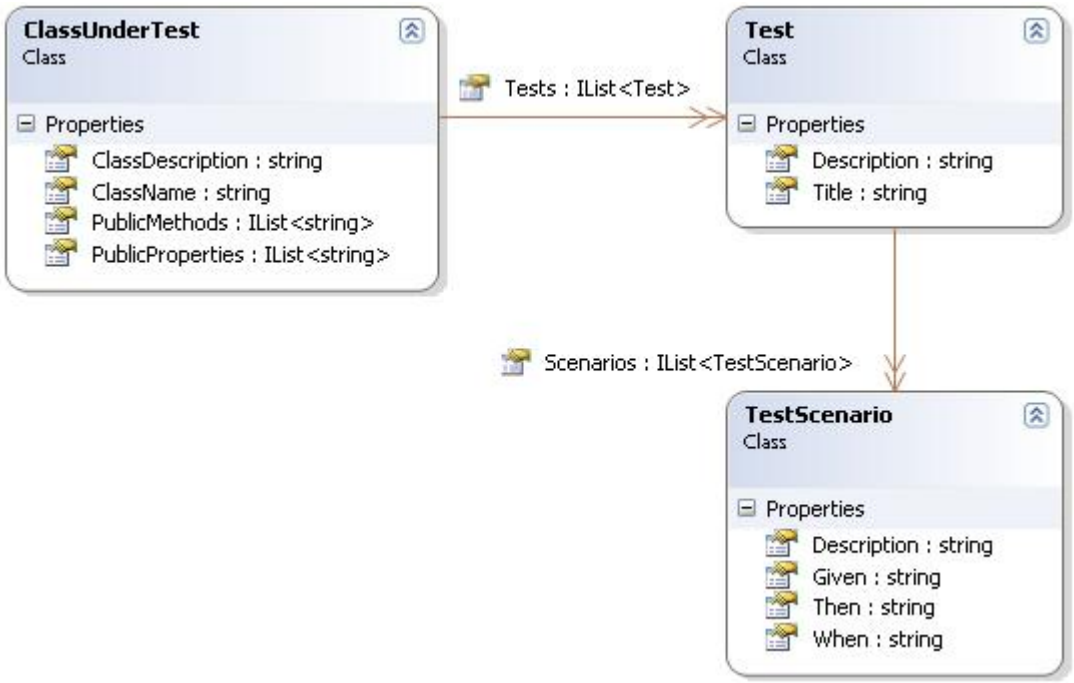

Figure 1:

the front end canvas.

Each 'ClassUnderTest' object can have one or more 'Test' objects. And a 'Test' object can have one or more 'TestScenario' objects. Developers must write their tests using the 'Given/When/Then'[6] technique to make tests descriptive.

- **Given:** This field provides a description of the initial state of the class and it's dependencies before the test is executed.
- **When:** This field contains a description of the event that occurs after the initial context was created.
- **Then:** This field provides us with the expected result(s).

This technique allows us to extract meaningful test metadata that provide tests with more context. This data can now be sent to any output device for consumption by users. In our system we use this data structure to generate a PDF file, we bind it to an HTML 5 web front end.

4 Methodologies and Approach

4.1 Project Management

As technical lead I have to work very closely to the project manager in the team. This is to ensure that we meet our delivery objectives on time. Please refer to Xoliswa's individual[7] report for more project management related information.

4.2 Development Methodology

The team used the Kanban methodology which falls under the 'Agile' umbrella of techniques. We used the free version of the Kanbanery tool which provides us with an online Kanban task board. Please refer to Xoliswa's individual report[7] for more on the development methodology.

4.3 Source Control

The team employed Git[8] as a source control management tool. Git is a free distributed version control management system. The team chose Github[9] as a central Git repository. Git was new to most members of the team at the beginning of the project. The team decided to use this opportunity and try out Git on this project.

Learning the Git commands was a bit of a learning curve at first, but the team quickly became comfortable with the utility.

This following link is point to the teams Github repository.
https://github.com/KeaganPhillips/Wit-Group-4-project

5 Conclusion

The solution developed gives programmers the ability to quickly glance over the system a view all test. This can be of great benefit to developers, as it gives them the full context of the internal working of a system (or a component within a system), this is especially true for new developers joining the development team as well as for junior developers allocated to the team.

The group hence felt that the unit test visualization project fulfill both stated requirements of software visualisation and/or education.

References

[1] Microsoft Dot Net Framework 4.0 http://www.microsoft.com/net

[2] Please see the following link for an in dept discussion on TDD (Test Driven Development) http://www.agiledata.org/essays/tdd.html

[3] For a detailed discussion on Pair Programming please visit http://c2.com/cgi/wiki?PairProgramming

[4] The unit test convention https://github.com/KeaganPhillips/Wit-Group-4-project

[5] Microsoft DotNet Reflection reference materiel http://msdn.microsoft.com/en-us/library/ms173183%28v=vs.80%29.aspx

[6] For more information on the Given / When / Then technique please visit http://testedobjects.sourceforge.net/m2-site/main/documentation/docbkx/html/user-guide/ch03s04.html

[7] Xoliswa's individual report https://github.com/KeaganPhillips/Wit-Group-4-project

[8] by Scott Chacon, Pro Git,published by Apress http://git-scm.com/book

[9] Gidhub source control provider https://github.com/

[10] b