The Way of Testivus

Less Unit Testing Dogma
More Unit Testing Karma

Good advice on developer and unit testing, packaged as twelve cryptic bits of ancient Eastern wisdom.

Translated by Alberto Savoia

www.agitar.com/downloads/TheWayOfTestivus.pdf

Translator's Introduction

In May 2006, an ill-prepared international expedition to the Himalayas lost its way. After two weeks of wandering around — hungry, thirsty, and smelling like inexperienced expeditioners who got lost for two weeks — they stumbled upon an ancient cave.

Inside the cave they saw a maze of cubicles. Each cubicle had a wooden desk, an ergonomically correct bamboo chair, a Dilbert™ calendar, and a strange computer-like mechanical device. In one corner of the office they found barrels of dark liquid (later identified as early examples of a carbonated and highly caffeinated drink) and a ping-pong table. They realized that the cave was an ancient software start-up. The oldest one on record. Older even than Netscape.

Among the many surprising things they discovered inside the cave was the most amazing one of all: a note left by one of the programmers. The expedition's guide, while not very good at guiding, knew how to read the ancient language and translated the note for them:

"We have finished the release ahead of schedule — again. All the tests pass, so we are taking the rest of the week off. We are going sailing. Since it's a team-building exercise, we hope we can get reimbursed for it."

The explorers looked at each other in astonishment. Not only had they discovered the oldest software start-up in history, they had also discovered a team of programmers who, apparently, completed their code ahead of schedule on a regular basis.

What was the secret of these ancient programmers? And what had happened to them? The expeditioners searched each cubicle for clues, and they found two well-worn booklets. One of them was called "Learn To Sail In 30 Minutes," which explained the fate of the programmers. You are holding in your hands a translation of the other booklet, "The Way of Testivus." Who wrote this mysterious booklet? What is Testivus? Only Google™ knows for sure.

Is the content of this text responsible for these ancient programmers being able to complete projects ahead of schedule? We can't be sure, but we believe that the amazing prowess of these programmers was probably due to a combination of the Testivus philosophy, and the consumption of large amounts of the dark caffeinated liquid found in the cave.

Read the booklet and draw your own conclusions.

Alberto Savoia, CTO/Cofounder of Agitar Software
April 2007, Mountain View, Calif.

If you write code, write tests

The pupil asked the master programmer:
"When can I stop writing tests?"

The master answered:
"When you stop writing code."

The pupil asked:
"When do I stop writing code?"

The master answered:
"When you become a manager."

The pupil trembled and asked:
"When do I become a manager?"

The master answered:
"When you stop writing tests."

The pupil rushed to write some tests.
He left skid marks.

If the code deserves to be written,
it deserves to have tests.

Don't get stuck on unit testing dogma

Dogma says:

"Do this.
Do only this.
Do it only this way.
And do it because I tell you."

Dogma is inflexible.
Testing needs flexibility.

Dogma kills creativity.
Testing needs creativity.

Embrace unit testing karma

Karma says:

"Do good things and good things will happen to you.
Do them the way you know.
Do them the way you like."

Karma is flexible.
Testing needs flexibility.

Karma thrives on creativity.
Testing needs creativity.

Think of code and test as one

When writing the code, think of the test.
When writing the test, think of the code.

When you think of code and test as one,
testing is easy and code is beautiful.

The test is more important than the unit

The pupil asked the great master programmer Flying Feathers:
"What makes a test a unit test?"

This great master programmer answered:
"If it talks to the database, it is not a unit test.
If it communicates across the network, it is not a unit test.
If it touches the file system, it is not a unit test.
If it can't run at the same time as any other unit tests, it is not a unit test.
If you have to do special things to your environment to run it, it is not a unit test."

Other master programmers jumped in and started arguing.

"Sorry I asked," said the pupil. Later that night, he received
a note from the grand master programmer. The note said:

"The answer from the great master Flying Feathers is an excellent guide.
Follow it, and most of the time you will do well.
But don't get stuck on any dogma.
Write the test that needs to be written."

The pupil slept well.
The other masters continued to argue long into the night.

The best time to test is when the code is fresh

Your code is like clay.
When it's fresh, it's soft and malleable.
As it ages, it becomes hard and brittle.

If you write tests when the code is fresh
and easy to change, testing will be easy,
and both the code and the tests will be strong.

Tests not run waste away

Run your tests often.
Don't let them get stale.
Rejoice when they pass.
Rejoice when they fail.

An imperfect test today is better than a perfect test someday

The perfect is the enemy of the good.
Don't wait for best to do better.
Don't wait for better to do good.
Write the test you can today.

An ugly test is better than no test

When the code is ugly, the tests may be ugly.

You don't like to write ugly tests,
but ugly code needs testing the most.

Don't let ugly code stop you from writing tests,
but let ugly code stop you from writing more of it.

Sometimes, the test justifies the means

The pupil asked two master programmers:
"I cannot test this code without mocking and violating encapsulation.
What should I do?"

One master programmer answered:
"Mocking is bad, and you should never violate encapsulation.
Rewrite the code so you can test it properly."

The other master programmer answered:
"Mocking is good and testing trumps encapsulation."

The pupil, confused, went out for a beer. At the local watering hole he saw the great grand master programmer drinking beer and eating buffalo wings.

"Great grand master," said the pupil, "I thought you did not drink.
And aren't you a vegetarian?"

The great grand master smiled and replied:
"Sometimes your thirst is best quenched by beer
and your hunger by buffalo wings."

The pupil was no longer confused.

Only fools use no tools

The farmer who does not use a plow
is not a good farmer.

The accountant who does not use an abacus
is not a good accountant.

Some tasks are best done with bare hands.
Other tasks are best done with tools.

It is not noble to do by hand
what can be done better with a tool.

It is not wise to use your head
when your head is not needed.

Good tests fail

The pupil went to the master programmer and said:

"All my tests pass all the time. Don't I deserve a raise?"

The master slapped the pupil and replied:

"If all your tests pass, all the time, you need to write better tests."

With a red cheek, the pupil went to HR to complain.
But that's another story.

The Way of Testivus

If you write code, write tests.

Don't get stuck on unit testing dogma.

Embrace unit testing karma.

Think of code and test as one.

The test is more important than the unit.

The best time to test is when the code is fresh.

Tests not run waste away.

An imperfect test today is better than a perfect test someday.

An ugly test is better than no test.

Sometimes, the test justifies the means.

Only fools use no tools.

Good tests fail.

This translation and printing of
"The Way of Testivus"
is brought to you by:

JUnit Factory

If you like the philosophy of Testivus,
and believe in less unit testing dogma
and more unit testing karma,
please visit:

www.JUnitFactory.com

JUnit Factory helps you increase your
unit testing karma by amplifying and
automating your testing efforts.

Best of all, it's free.