Queensland University of Technology

CAB402 - Assignment 1

Report

Name : Jamie Martin (N10212361)

Due Date : 23rd April 2021 [11.59 p.m]

Table of Contents

Table of Contents **1**

1.0 Introduction **2**

2.0 Discussion **2**

2.1 Efficiency 2

2.2 Effectiveness + Usability 2

2.3 Legibility 3

2.4 Maintainability 3

2.5 Conciseness 3

3.0 Conclusion **3**

1.0 Introduction

This report delves into a comparison between the two approaches (F# pure functional and C# object-oriented imperative) taken for a Genetic Algorithm solution, both in terms of the computational efficiencies as well as the development experience along the way.

2.0 Discussion

Functional programming still seems to come second to imperative languages despite the benefits it may hold. Though the way in which developers are taught may play a role in this, as very few begin learning with functional programming. This could potentially be creating a barrier of entry for developers unbeknownst to functional programming, as there are many concepts never seen before, nor of common knowledge.

2.1 Efficiency

Despite the use of higher-order functions in the F# implementation, the efficiency of the execution did not fall short compared to the C# implementation. Over a number of tests the C# implementation averaged 36 seconds, as opposed to 33 seconds for the F# implementation. One would assume, due to the abstraction of these higher-order functions, it might be easy to over-utilise these abstractions which may lead to code that might not perform particularly well, despite being more concise. Due to the abstraction being hidden it is easy to overlook its implementation; however, this was not the case.

2.2 Effectiveness + Usability

Despite F# having a slight learning curve to begin with, it was rather easy to be productive with it. The use of higher-order functions, pipeline operators, and currying, enabled quicker development times allowing for abstraction of otherwise simple, yet time-consuming methods as seen in the C# implementation.

2.3 Legibility

The use of higher-order functions and pipeline operators in the F# implementation, creates a more concise, holistic piece of code. With this, it can mean that the code implemented is more legible, as there is an understandable flow to the execution and abstractions that are commonly understood by developers are hidden.

2.4 Maintainability

Due to the concisity of the F# implementation, one can derive that the maintainability follows suit; as in fewer lines results in more maintainable code. With the F# implementation being only 200 odd lines, one can see how it would be much easier to maintain. This can also be said likewise for other developers looking at the implementation. Even if they do not have a solid grasp of F#, it is ultimately fewer lines to have to make sense of, and is therefore more maintainable.

2.5 Conciseness

A basic line count comparison paints the whole picture. With the F# implementation amounting 167 lines, compared to the C#'s 389, it is easy to deduce the winner in terms of conciseness. This is predominantly due to the use of higher-order functions.

3.0 Conclusion

F# has been a pleasure to work with - once the initial hurdles had been made. The conciseness of implementations and abstraction through higher-order functions enables the developer to focus solely on the domain logic; propelling forward the productivity one can have. As the saying goes "Less is more", and that certainly rings true comparing these two implementations. The fact that were seeing the use of System.LINQ more and more in the C# Community is a testament to the effectiveness functional approaches have.