**Efficient Task-Based Solution for a Given Software Engineering Problem That Uses Parallelism.**

**Hand in deadline: Friday 20th March 2015 before 3:30pm**

**Report**

**By Giovanni Lenguito**

**Contents**

[Introduction 3](#_Toc414294432)

[Technology 3](#_Toc414294433)

[Extra Features 4](#_Toc414294434)

[Performance Evaluation 4](#_Toc414294435)

[Testing 5](#_Toc414294436)

[Attachments 6](#_Toc414294437)

# Introduction

In this document I will explain the system design, diagrams and performance evaluation. The software developed is for Spur. At the end of this documentation you will find the testing that was produced. List of extra features will be mentioned in this document also. The application fulfils the user’s requirements which were:

* A Graphical User Interface using WinForms or WPF
* All features should be calculated when the option is selected in the UI, not at initial load. This will allow the data sets to be updated and the calculations rerun without reopening the application. Spur's technical team would like the UI to continue being responsive during these calculations.
* Have the flexibility to point the application to a certain folder on a PC to find the .csv files.
* List all stores, suppliers and supplier types.
* Allow the Finances team to find the following data:-
  + The total cost of all orders available in the supplied data
  + The total cost of all orders for a single store
  + The total cost of orders in a week for all stores
  + The total cost of orders in a week for a single store
  + The total cost of all orders to a supplier
  + The cost of all orders from a supplier type
  + The cost of orders in a week for a supplier type
  + The cost of orders for a supplier type for a store
  + The cost of orders in a week for a supplier type for a store

The ability to plot the historical supplier order data on a graph has currently not been implemented. This function will follow in future updates to the Spur Finance Application.

# Technology

I developed the application in C# using WPF over Windows forms. This is because WPF is more powerful and is the latest technology. The application uses TPL (Task Parallel Library) in order to get the best performance form dealing with a large amount of data. Using tasks prevents the user interface from freezing up and hanging. I will be using Microsoft Visual Studio 2013 Professional 2013 as my choice of IDE to develop the application.

Started of using WinForms to get the basic functionality and design for the user interface then promptly more onto WPF. WPF allowed me to develop a more attractive, effective and responsive user interface.

# Extra Features

Some of the extra features added were designed to help speed up productivity within the application. For example a settings windows has been added which allows the user to save paths to the folder and stores file. This allows the user to click a button to load the data and auto load stores. There is also an option to toggle between Parallel and Sequential code. To help the user understand the application there is a welcome screen which can be toggles on or off on start up. The welcome screens tells the user what to do and how to use the application.

# Performance Evaluation

On the very first time you load the application the time it takes to load and display the folder selected it takes around 30 seconds. After this every other attempt takes around 7 seconds. The cause of this is currently unknown but I might be down the allocation of threads. The option to switch between parallel and sequential is built in the settings of the application. This will allow you to see the speed difference between the two. Below is a table showing the times taken to load and display the folder selected.

|  |  |  |
| --- | --- | --- |
| Attempt | Parallel | Sequential |
| 1 | 00:00:05.1423689 | 00:00:07.6220202 |
| 2 | 00:00:06.0378696 | 00:00:10.9925419 |
| 3 | 00:00:05.9658013 | 00:00:10.3916339 |

As you can see from the result there isn’t too much of a difference however it is a performance increase! There is around a 4 second increase. Collections used are Concurrent Queues and List. Using Concurrent Queues I found there was less overhead and I could quickly get data out of it.

When the user selects a folder and clicks ‘Okay’ the application will go through all the files inside that folder only if the file has the extension .csv. Each file will be parsed. The individual data gets turned into an object which is stored in a collection. In this instance the collection I have found to be the fastest to access is the concurrent queue. Two hash sets are set up to hold the data for the supplier and supply type. This is then used to populate the filters to generate the report. Final thing that happens is that all the data prints to the data grid for the user to review.

The time it takes to generate the report is very small. On average it takes 2 seconds. In the application the thing I found that takes the most amount of time is the printing the data to the screen using the GUI thread. To overcome this issue I generated everything in a separate thread ready for the dispatcher to call and display to the user.

The speed of the application mainly depends on the end users computer hardware. Running on 4 core processor is no problem, I have not tested the application on a 2 core processor. The program should still work however it will take a longer time to read the data and parse.

Using Concurrency Visualiser I could see the application was using a lot of threads. This was due to using parallel and tasks. I also noticed there was a lot of background tasks running on my machine which meant I couldn’t fairly test my application’s time.

# Testing

The data the finance team needs works and functions correctly. Below is a test of the application working.

|  |  |  |  |
| --- | --- | --- | --- |
| Screenshot | Supposed to happen | Happens? | Fix |
|  | Welcome windows pops up if show welcome is true, else don’t show | True | N/A |
|  | ‘Select Folder’ button opens explorer window and allows user to select a folder | True | N/A |
|  | When folder is selected data will load, populating combo-boxes and data grid. As well as update status bar | True | N/A |
|  | ‘Load Stores’ button opens explorer window and allows user to select a file named StoreCodes.csv | True | N/A |
|  | When file selected and ok is clicked, the combo-box will populate with the data in the csv file | True | N/A |
|  | If ‘Generate’ button clicked and no filters have been selected message box will appear | True | N/A |
|  | If filters have been selected and ‘Generate’ button has been clicked the data grid will populate selected items and the statistics will update on the right, as well as the status bar. | True | N/A |
|  | If ‘Calculator’ button clicked the windows calculator application will pop up. | True | N/A |
|  | If ‘Settings’ button is clicked the settings windows will pop up and display saved data from previous settings made. | True | N/A |
|  | Select path on both textboxes will pop up explorer to allow the user to select their paths. On save the data is saved even when the application is closed. | True | N/A |

# Attachments

* L010516C\_Activity\_Diagram.vsdx
* L010516C\_User\_Case.vsdx