Final Year Project

Energy Consumption of Design Patterns

Anna Davison

Student ID: 16382333

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Dr Mel Ó Cinnéide

UCD School of Computer Science

University College Dublin

May 19, 2020

Table of Contents

Abstract

Energy Consumption is a property of the programs generated by software developers. The extensive reliance on enterprise applications by organisations to run their businesses along with a proliferation of end-user applications accessible through mobile computing has created an exponential growth in energy consumption through data centres and mobile technology. Climate change is now a reality. The need for the world to switch to clean energy and at the same time reduce energy consumption is increasingly important if the worst excesses of climate change are to be avoided. Recent developments in software design have placed significant reliance on the use of Design Patterns for speed of development, ease of maintenance and code reuse. The hypothesis of this project is that Patterned and non-Patterned programs have different profiles of energy consumption, that is, programs utilising Design Patterns are less energy efficient. This hypothesis was tested using state-of-the-art tools to measure and compare static energy efficiency and run-time energy consumption of a sample of Patterned and non-Patterned Java programs.

Outline of Report Structure

Interim Report Sections:

1. Abstract
2. Project Specification
3. Introduction
4. Related Works and Ideas
5. Experimental Approach
6. Project Work Plan
7. Statistical Planning
8. Summary and Conclusions
9. Acknowledgements
10. Bibliography

Additional Final Report Sections:

1. Key Learnings
2. Evaluation

Modifications

1. Updated report structure to reflect new structure
2. Updated experimental framework to reflect the disuse of safe mode
3. Updated data collected section to reflect what occurred
4. Updated breakdown of tasks to reflect what occurred
5. Added new Gantt Chart to chapter four
6. Updated chapter five to reflect change of chosen statistical tests
7. Updated the Summary and conclusion to reflect the results
8. Added an introduction to each chapter

Project Specification

Core

Investigate the state of the art in tools to measure static energy efficiency and run-time energy consumption of Java programs, and decide what tools to use for this project.

Select three standard Design Patterns and for each one implement several pairs of applications: one that uses the pattern and an equivalent one that does not use the pattern. Several pairs will be required as a pattern has many possible variants. Create a typical load for each pair and compare the energy efficiency of the Pattern version and the non-Pattern version of each program.

Note that this investigation is a formal experiment testing the hypothesis that the energy of the pattern and non-Pattern versions have different profiles of energy consumption. The energy consumption of a program for a given load is a range of values, and the appropriate statistical tests should be used to test the hypothesis.

Advanced

Experiment with further patterns, and try to determine if certain micro-patterns (e.g. delegation, inheritance) are the source of the energy inefficiency.

Experiment with a pattern such as Object Pool that typically (and unusually, for a Design Pattern) yields performance improvement, and test if it also yields a commensurate improvement in energy efficiency.

Chapter 1: **Introduction**

This chapter introduces the importance of this project and why it is worthy of research and investigation. Section 1.1 reflects on the importance of energy consumption and the use of Design Patterns in IT. Section 1.2 considers the potential real-world impact of the project on energy consumption, refactoring and on Design Patterns in computing.

1.1 The Importance of Energy Consumption and Design Patterns in IT

The recent exponential growth of computing services through the wide-spread adoption of the end-user computing devices, often battery-powered. This has driven demands for larger batteries and longer running times as well as greater processing efficiency. At the same time, the world is grappling with climate change and the impact of consuming fossil fuels to produce electricity for the powering of these devices. With mass adoption resulting in micro applications being used by billions of users [1], the software industry can no longer be complacent about the energy efficiency of software used by these devices. While minor energy consumption improvements may make only minor differences to individual devices and users, the impact of a small improvement on many millions of users can no longer be ignored.

Software Design Patterns were developed from the traditional concept of patterns (e.g. knitting patterns, sewing patterns). Patterns were introduced originally to help architects design buildings. [2] The concept inspired developers to create repeatable patterns for software so that they too could have a Template Method to follow in regularly used design situations. Design Patterns were created to enable developers to efficiently develop code which is easily reusable and well documented. With growing concerns about climate change and the world's heavy dependence on fossil fuels to generate electricity [3], this is a topic which can no longer be taken for granted.

With a growing concern for the energy consumption of micro computing services, an understanding of the impact of the use of software components such as Design Patterns on energy consumption is essential to inform and guide future development decisions. This project seeks to determine if the use of Design Patterns increases or reduces energy consumption in software programs. The hypothesis is that the use of Design Patterns increases energy consumption.

1.2 Real-World Impact of Project

1.2.1 Energy Consumption

All technology utilises energy to fulfill its function. As a result of this, and the rapidly expanding use of technology, determining how to save energy is of the upmost importance. Software is becoming increasingly complex and requiring access to more power demanding infrastructure. With growing concerns about climate change and the growth of power-hungry data farms this is an issue that the software industry needs to address.

In Ireland, it has recently been announced that two peat-burning power stations are to be shut down. [4] At the same time the Irish Government has committed to generate 70% of the country's electricity supply from renewable sources by 2030. [5] These decisions of switching away from fossil fuels to renewables are mirrored in many other countries around the world - meanwhile it is projected that world energy consumption will increase by nearly 50% by 2050. [6] Contributing to this growth in energy demand is our increasing use of technology and software in our everyday lives. The building of power stations, wind farms and solar farms are multi-year projects while the re-coding of software to be more energy efficient can be a relatively straight forward task. Due to its moderate climate Ireland has a particularly high level of data farms - it also has high levels of emissions of green house gases. In an age when brand value can be wiped out in an instant with a simple tweet, developers of software need to be seen to be playing their part to minimise their impact on climate change. By optimising code to minimise energy consumption they will be enhancing the public image of their organisation while playing their part to minimise the impact of climate change.

1.2.2 Refactoring

Refactoring is the task of restructuring a piece of code which involves improving the readability and reducing the complexity of the code without changing its function. Refactoring of software is one of the best ways to optimise application performance. If it is determined that code that utilises a Design Pattern does consume more energy, then we could consider developing an automated refactoring process within a compiler that "de-patterns" the software as the executable is created. Therefore we may still benefit from the use of Design Patterns without the associated increased energy costs.

1.2.3 Design Patterns

Design Patterns are template solutions used to address common software design problems. They are used to improve readability, flexibility and reuse of software. However, some of these Design Patterns are computationally complex and could consume more energy than software that does not use these patterns. That is why refactoring could be a possible solution to this problem as mentioned above. Before we consider whether or not there is a need for this tool however it is necessary to investigate if there is a increase in energy consumption as a result of the use of Design Patterns.

1.3 Report Structure

Chapter two considers related work and how it affects this project. Chapter three describes the experimental approach for the project. Chapter four details the project workplan and how the project actually occurred. Chapter five sets out the evaluation plan for the data. Chapter 6 outlines the choices that were committed to or rejected. Chapter seven details the analysis of the results. Finally, chapter eight summarises and concludes this report.

Chapter 2: Related Work and Background

This chapter reviews the background and basis of this project together with the tools used. Section 2.1 considers the definition of energy and how it is consumed by a computer. Section 2.2 reviews the history of Design Patterns and relevant papers about their energy consumption. Section 2.3 discusses the different methods of measuring energy consumption that were investigated and considered.

2.1 Energy Consumption of Software

Energy is the capacity for doing work. [7] Energy cannot be created or destroyed, only converted from one form to another. [7] Energy comes in many forms such as potential (energy that is stored in a object) to kinetic (energy of moving objects) to thermal (heat energy). [8] Power stations create electricity from the burning of fossil fuels, nuclear fusion, hydro, wind etc.

Computers have many components that consume energy: the CPU, hard drives, graphic cards, monitor and speakers to name a few. The rate at which components use power depends on what tasks they are performing and what state the computer is in. For example, if the computer was running a game that is hosted online and utilises a high frame rate, this would require a high energy level due to the need to refresh the screen frequently to ensure a smooth video presentation. This would be very demanding on the graphics card. In comparison, if the computer was in sleep mode, all programs would be set to idle which would use significantly less energy than an online game.

Software design impacts the energy consumption of a system. [9] The more complex and large a system is, the more likely it will consume more power. The paper cited above measures the energy consumption of six C++ programs, before and after the appropriate design has been applied. The researchers were conducting similar tests to determine if the use of Design Patterns increased or decreased the energy consumption. The researchers concluded that Design Patterns do not notably increase energy consumption however more research is required in order to validate the conclusion by testing a greater range of Design Patterns. As each Design Pattern was tested only once for the purposes of this research, the results and conclusions are are subject to more extensive validation. In this project each Design Pattern will be tested three ways in order to take into account variants within the code. The researchers calculated the difference in energy consumption with percentages - in this project universally recognised statistical calculations will be used to assess the hypothesis.

2.2 Design Patterns

In 1977/78, A civil architect named Christopher Alexander instigated the idea for patterns. [2] His patterns related to civil architecture but they inspired developers from the object-oriented programming (OOP) community to develop the concept for programming. Developers of note being Kent Beck and Ward Cunningham who presented their ideas at an OOPSLA conference. This led to their ideas spreading among the OOPs community. Eventually their ideas became world-known and widely used, as they are to this day.

A Design Pattern is a broadly reusable solution to a problem that is regularly encountered in software design. [2] Design Patterns are created by experienced developers as templates that can be applied to specific situations. A Design Pattern has four components: its name, the problem it applies to, the solution to the appropriate problem and the impact of the pattern on the program. We use Design Patterns because they are flexible, reusable, maintainable, easy to read and follow best coding practices.

The difference in energy consumption between Patterned and non-Patterned programs can vary significantly from pattern to pattern. [10] From this paper the researchers state that there may be a greater difference in energy consumption between Patterned and non-Patterned versions of an application due to the increased number of object instantiations and method calls but were unable to reach this conclusion without further research.

This paper measures the energy consumption of six smart-phone applications, before and after the appropriate Design Pattern have been applied. They concluded that Design Patterns do increase energy consumption. Of the Design Patterns tested, the Decorator Design Pattern was found to have the greatest differential in energy consumption. The researchers also concluded that although they obtained clear results from their tests, further research is required to generalise the results from these tests. This paper was more persuasive than the Litke paper [9] as it tests five Design Patterns instead of three, however the researchers only tested each Design Pattern against one application which does not take into account design variability. As was stated previously, this project will test each pattern with three different applications, which should help combat the variability within Design Patterns. This paper also calculated the difference in energy consumption utilising percentages. As stated previously, this project will use statistical tests to measure the difference in energy consumption between applications.

2.3 Methods of Measuring Energy Consumption

Energy can be measured in many forms. A Watt is a measure of electric energy for a certain amount of time. A Watt is a standard unit of energy equivalent to one Joule per Second. [11]One Joule is equal to the amount of work done by a force of one Newton that moves an object by one meter. A Newton is the force necessary to provide a mass of one kilogram with an acceleration of one metre per second per second. [12]

Energy consumption can be measured indirectly or directly. [13]

The indirect approach to measuring energy consumption is by inferring the energy consumption from other events. [13] For example, by examining the number of data cache-dependent stall cycles in a pipeline - using regression analysis we can determine the correlation between certain events and energy waste. A predictive model can then be created from this. This approach will not be utilised in the project.

Using the direct approach, energy can be measured with a monitoring instrument which can be installed within the device, in this case a personal computer. As software developers do not always have the ability to adjust their hardware, this method will not be used for this project. This project will instead use external monitoring hardware as it is easier for software developers to connect and utilise and has a shorter learning curve.

2.3.1 State-of-the-Art Review of Tools Considered for Direct Energy Measurement

The following tools have been considered for the direct measurement of energy consumption.

- MAchine Guided Energy Efficient Compilation Wand
- Generalised Target Architecture
- PowerTutor App
- Watts Up? Pro

This list of tools was determined from a review of: projects that were of a similar nature to this, open-access peer-reviewed papers and the recommendations of this projects supervisor.

In assessing these tools, the following criteria were considered: cost, developers knowledge, ease of use and devices capability's.

MAchine Guided Energy Efficient Compilation Wand [13]

This project was created to optimise compilers to save energy. The hardware and software involved in this project is open-source. Using their designs, anyone with the appropriate technical ability can build their own wand. The wand was created over the course of 2012/13. The picture below (Fig.1) is of the MAGEEC wand attached to an STM$_{32}$F$_4$ discovery board. The wand has several features. It can measure the energy consumption of three channels at the same time. Each channel is adjustable to measure a range of different input values. The wand has a sample rate of up to two million samples per second. The wand also has a Graphical User Interface (GUI) when connected to an appropriate device. It has been successfully used to measure the energy of a diverse range of embedded systems. This technique may not be the most appropriate method to measure the energy consumption of the applications in this project as many developers have no experience in the physical hardware to build this device.

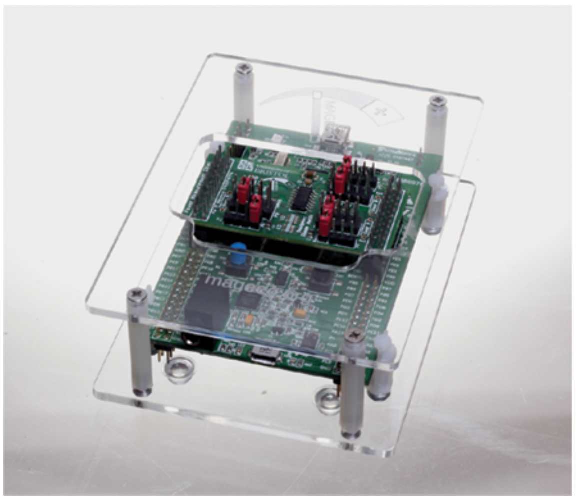


Fig.1. MAGEEC wand attached to an STM$_{32}$F$_4$ discovery board. Source: [13]

Generalised Target Architecture [9]

In this paper the researchers used an ARM7 integer processor core because of its high MIPS (Million Instructions Per Second) performance. It was the most widely used architecture in mobile-devices as of 2011. [14] Using the processor core the researchers measured the number of cycles, the opcode memory accesses, the data memory accesses, the processor energy, the instruction memory energy and the data memory energy. The researchers then calculated the percentage difference between the Patterned and non-Patterned applications for each of these data points. They concluded that Design Patterns do not in general increase energy consumption significantly but that further research is required. As mentioned in section 2.1, this is not the most appropriate method to test the energy consumption of Design Patterns as the researchers used percentages instead of statistical tests. It also appears that the process of obtaining the values required to measure the energy difference in applications is extremely arduous and is not recommended. Ultimately, the values presented in the research do not show how much energy the applications consumed.

PowerTutor App [10]

This app was released in 2009. This app is unsuitable for this project as we are not specifically testing the energy consumption of mobile apps. We are testing the energy consumption of Java apps executed on a computer.

Watts Up? Pro [15]

This is an external commercially-available device. According to its manual it was launched in 2003. [16] The company that produces this device is no longer in business. It utilises automated recording which is similar to how breathalyzers operate. This product measures the energy consumption of objects that source their energy from electrical sockets. In this paper the researchers have tested the meter to determine its reliability. The device can measure energy in Watts (among other metrics), record data, record energy in increments of seconds, can produce graphs and can transfer data via USB. The meter can store up to 32,000 records. The data can be exported to Microsoft Excel. The manufacturers report the accuracy of the device is in the range of plus or minus 1.5% + 0.3 Watts. The researchers conclude that the device is sensitive to changes in energy consumption and that it is reliable. The device does have limits however. It is not reliable when recording a low measure of power. The manufactures state that the device is inaccurate when measuring values less than 0.5 Watts. This paper states that any measurement that is recorded that returns a value lesser than or equal to one Watt is to be regarded as generating zero Watts to avoid reliability issues. In summary, this paper recommends the use of this device due to its many features and relatively low cost. This device is therefore considered suitable for the purpose of this project. Fig.2 below is a picture of the Watt Up? Pro. As it is made for use in America this project will be using adaptors in order to use the device with the local power supply and the computer upon which the applications will be executed.

Fig.2. Watts Up? Pro meter used to measure energy consumption.

Source: [17]

Chapter 3: Experimental Approach

This chapter introduces the testing methodology utilised in the project, the Design Patterns that were tested and the type of data collected from those tests. Section 3.1 describes the testing methodology. Section 3.2 discusses the Design Patterns that were tested. Section 3.3 describes the type and extent of the data that was collected during each test.

3.1 Framework

In order to measure the energy consumption of the applications it was determined that Watts Up? Pro was the best device to utilise. The applications were run on the command prompt of the PC. The computer that was used in this project is a HP Envy Notebook with an Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz with 8GB of RAM and a 64-bit operating system.

The testing methodology is as follows:

1. Boot the computer and log on.

2. Open a Command Prompt.

3. Compile the application that is being tested.

4. Plug the computer into the Watts Up? meter (with an adaptor).

5. Plug the meter into the power outlet (with an adaptor) - the meter will start automatically and begin logging data.

6. Wait five minutes so that the meter can measure the idle energy consumption of the computer.

7. Execute the application.

8. Once the application has completed, unplug the meter.

9. To analyse the results, connect the meter to the computer using the supplied USB cable.

10. Download, analyse and interpret the recorded data.

3.2 Design Patterns

All the Design Patterns that were tested are from the Behavioral Design Pattern category. Behavioral Design Patterns are concerned with identifying communication patterns between objects and classes. [2] The following Design Patterns within this category were tested: Strategy, Template Method and Observer.

3.2.1 Design 1 - Strategy

The Strategy Design Pattern, as illustrated in Fig.3 below, is used when there is a group of connected algorithms and a client object needs to be able to choose which algorithm it wishes to use based on what it wishes to accomplish.

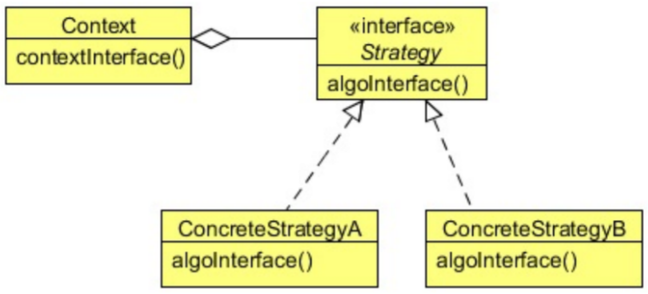


Fig.3. Structure of Strategy Design Pattern. Source: [2]

3.2.2 Design 2 - Template Method

The Template Method Design Pattern, as illustrated in Fig. 4 below, is used to define the skeleton of an algorithm. It can be used when there is an algorithm that can be implemented in a variety of ways. The template class holds the outline for this algorithm and the parts of the algorithm that are constant (the invariant). The subclasses hold the part of the algorithm that is particular to that method of solving the problem (the variant). The Template Method Design Pattern removes the need for code replication, which helps ease the maintenance of the code and reduces the amount of code that is to be written. This ensures consistency in the algorithm where it is needed.

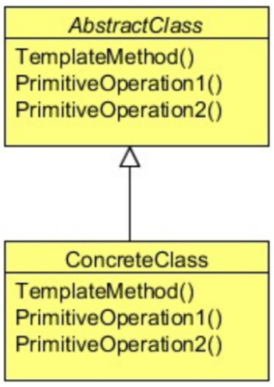


Fig.4. Structure of Template Method Design Pattern. Source: [2]

3.2.3 Design 3 - Observer

The Observer Design Pattern, as illustrated in Fig. 5 below, is used to define a one-to-many dependency between objects. When one object changes, its dependents are automatically notified and updated. The Design Pattern incorporates:

- a Subject class which is used to add and remove observers.
- an Observer class that gets updated when the subject changes, it also implements the Observer interface.
- a Concrete Subject class that implements the subject interface and it updates all the observers when there is a change in the subject.
- a Concrete Observer class that implements the observer interface and it contains a reference to a concrete subject from which it receives changes in state.

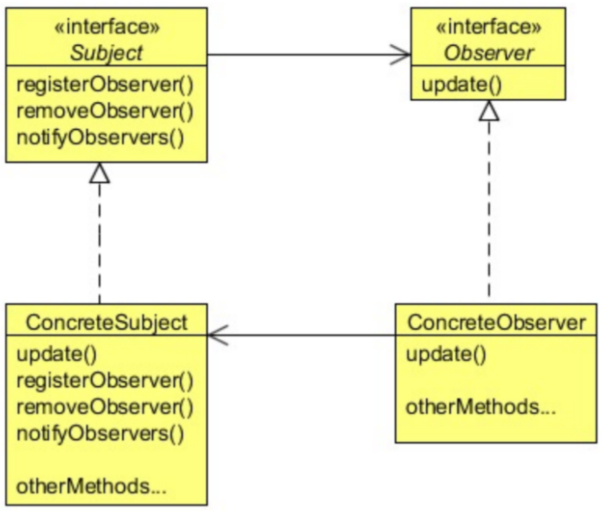


Fig.5. Structure of Observer Design Pattern. Source: [2]

3.3 Data Collected

In total three applications were tested. Patterned and equivalent non-Patterned versions of the applications were each tested. There are three Design Patterns, each Design Pattern was tested with an application. The Watts Up? meter was used to measure energy consumption in Watts. Each of these applications was tested as laid out in the testing methodology. Each time an application was tested the average idle Watts was measured prior to executing the program as well as the average Watts while the program was executing. The difference between these values was the energy consumed by the program. Each application was tested thirty times. This generated thirty values from the Patterned application and thirty values from the non-Patterned application. These two sets of values were used in statistical tests, as laid out in chapter five.

Chapter 4: Project Workplan

This chapter sets out the breakdown of tasks undertaken in this project and the Gantt Charts that lay out the predicted and actual timelines of the work completed. Section 4.1 lists the tasks involved in this project. Section 4.2 presents the Gantt Charts that show the predicted and actual duration of the project and the task dependencies.

4.1 Breakdown of Tasks

1. Find an application with the Strategy Design Pattern. Remove the Design Pattern from that application to obtain the non-Strategy equivalent of that application.

2. Test both applications as laid out in the testing methodology.

3. Use statistical tests as detailed in chapter five to interpret the recorded data.

4. Find an application with the Template Method Design Pattern. Remove the Design Pattern from that application to obtain the non-Template Method equivalent of that application.

5. Test both applications as laid out in the testing methodology.

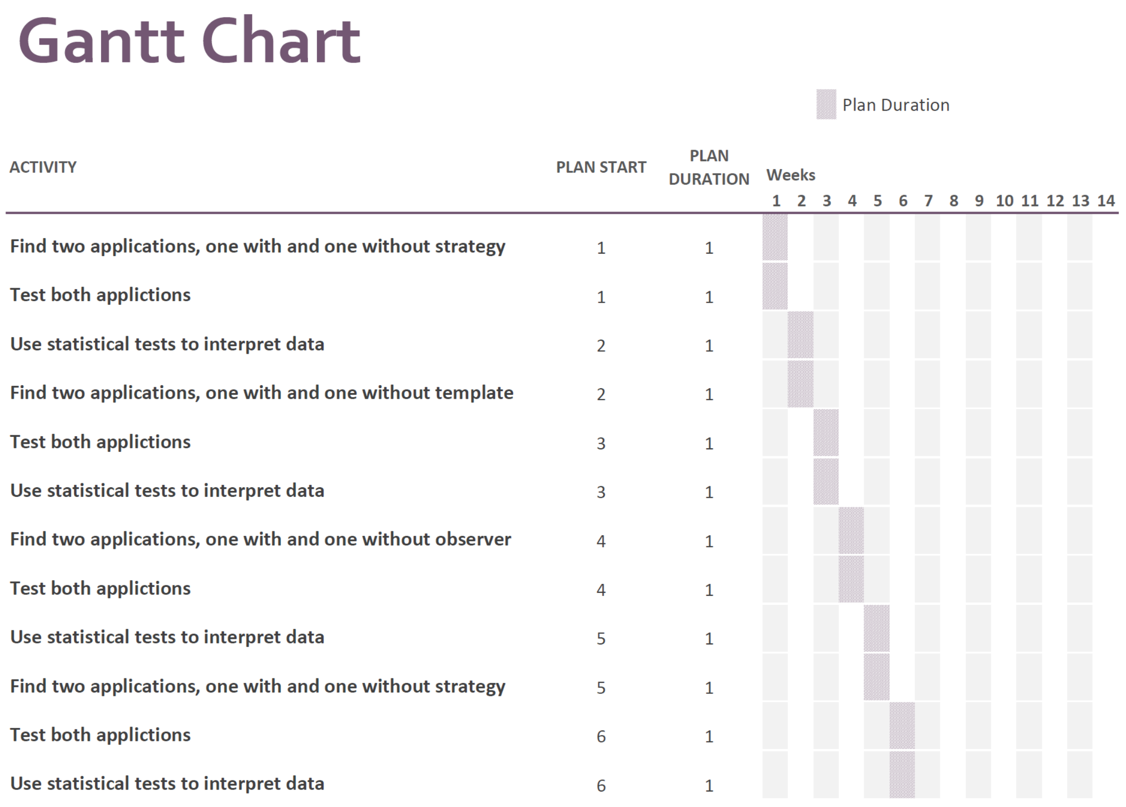
6. Use statistical tests as detailed in chapter five to interpret the recorded data.

7. Find an application with the Observer Design Pattern. Remove the Design Pattern from that application to obtain the non-Observer equivalent of that application.

8. Test both applications as laid out in the testing methodology.

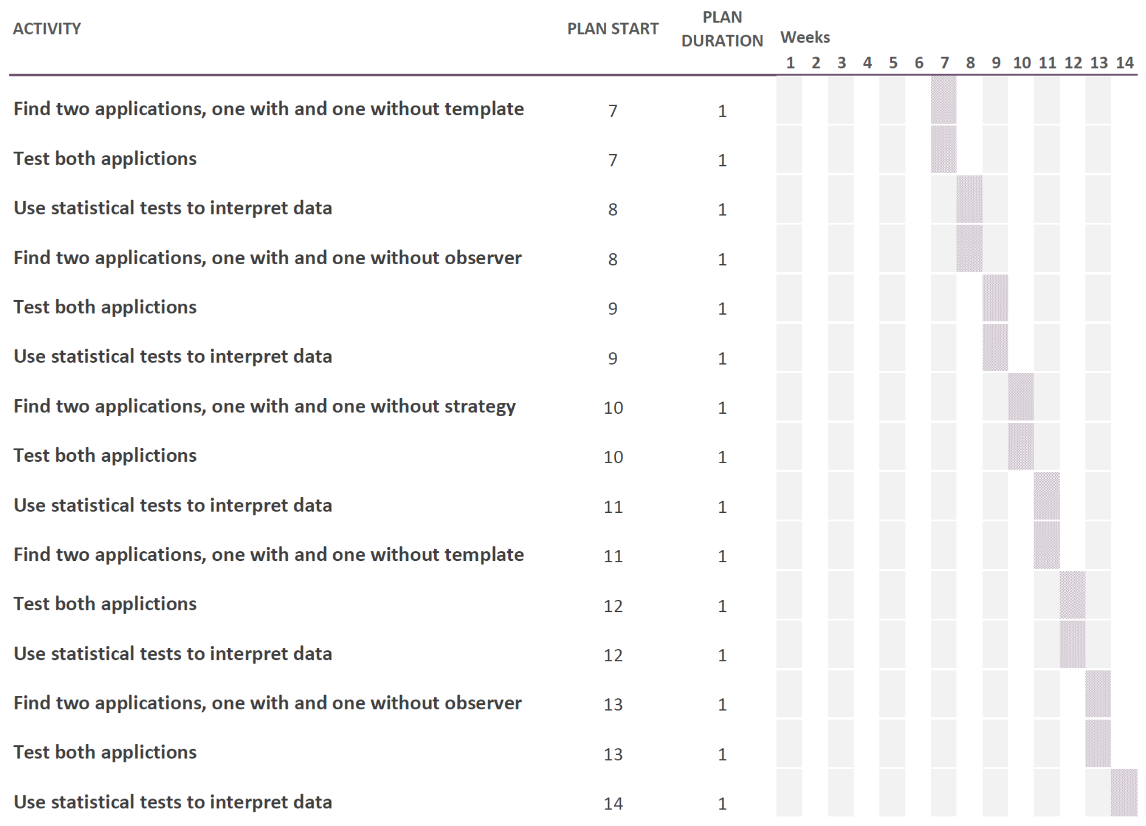
9. Use statistical tests as detailed in chapter five to interpret the recorded data.

4.2 Gantt Charts


Gantt Chart

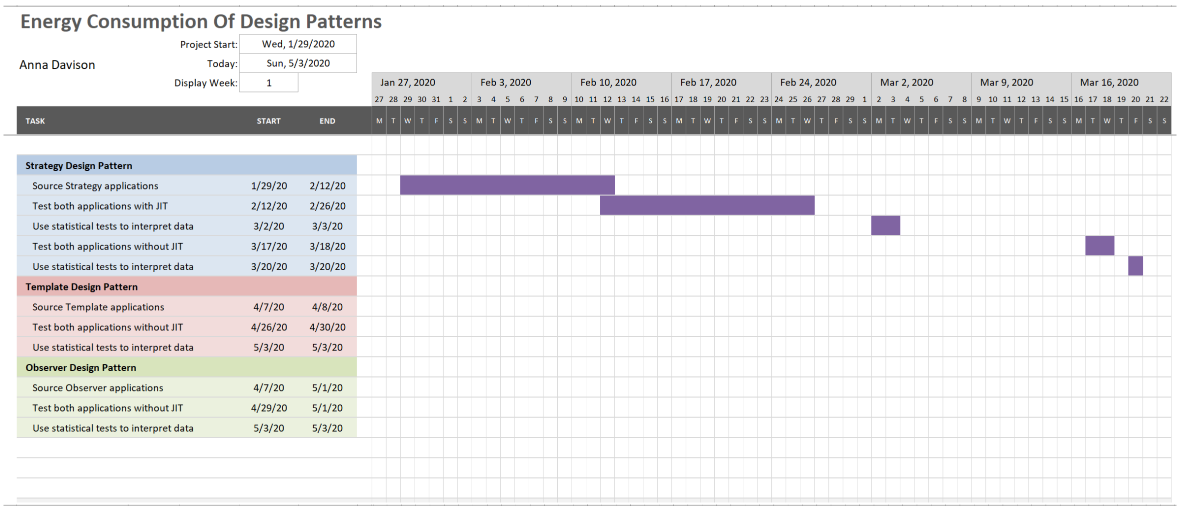
▨ Plan Duration

ACTIVITY	PLAN START	PLAN DURATION	Weeks 1	2	3	4	5	6	7	8	9	10	11	12	13	14
Find two applications, one with and one without strategy	1	1	▨													
Test both applictions	1	1	▨													
Use statistical tests to interpret data	2	1		▨												
Find two applications, one with and one without template	2	1		▨												
Test both applictions	3	1			▨											
Use statistical tests to interpret data	3	1			▨											
Find two applications, one with and one without observer	4	1				▨										
Test both applictions	4	1				▨										
Use statistical tests to interpret data	5	1					▨									
Find two applications, one with and one without strategy	5	1					▨									
Test both applictions	6	1						▨								
Use statistical tests to interpret data	6	1						▨								



ACTIVITY	PLAN START	PLAN DURATION	Weeks 1	2	3	4	5	6	7	8	9	10	11	12	13	14
Find two applications, one with and one without template	7	1							▨							
Test both applictions	7	1							▨							
Use statistical tests to interpret data	8	1								▨						
Find two applications, one with and one without observer	8	1								▨						
Test both applictions	9	1									▨					
Use statistical tests to interpret data	9	1									▨					
Find two applications, one with and one without strategy	10	1										▨				
Test both applictions	10	1										▨				
Use statistical tests to interpret data	11	1											▨			
Find two applications, one with and one without template	11	1											▨			
Test both applictions	12	1												▨		
Use statistical tests to interpret data	12	1												▨		
Find two applications, one with and one without observer	13	1													▨	
Test both applictions	13	1													▨	
Use statistical tests to interpret data	14	1														▨

Gantt Chart of how the project was predicted to proceed

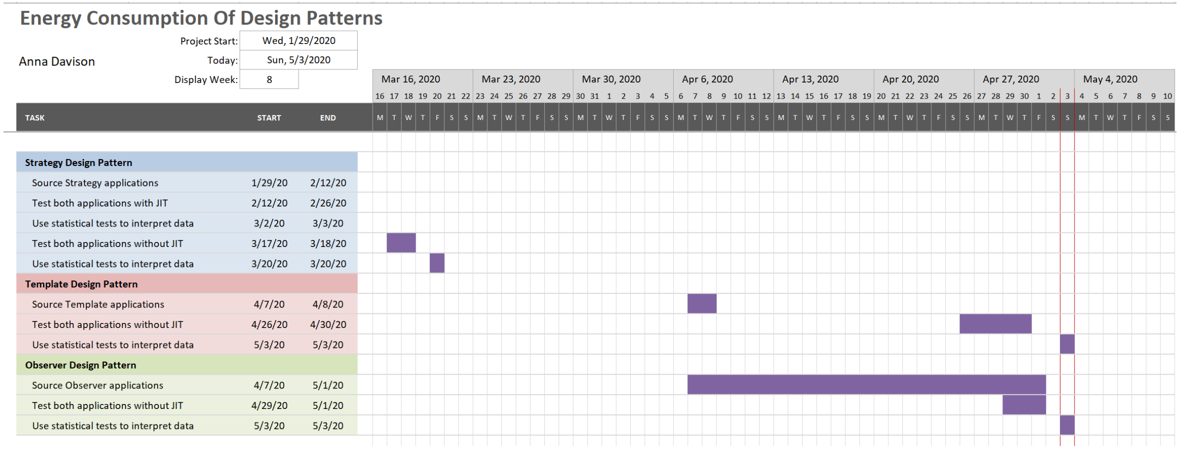


Energy Consumption Of Design Patterns

Anna Davison

Project Start:	Wed, 1/29/2020
Today:	Sun, 5/3/2020
Display Week:	1

TASK	START	END
Strategy Design Pattern		
Source Strategy applications	1/29/20	2/12/20
Test both applications with JIT	2/12/20	2/26/20
Use statistical tests to interpret data	3/2/20	3/3/20
Test both applications without JIT	3/17/20	3/18/20
Use statistical tests to interpret data	3/20/20	3/20/20
Template Design Pattern		
Source Template applications	4/7/20	4/8/20
Test both applications without JIT	4/26/20	4/30/20
Use statistical tests to interpret data	5/3/20	5/3/20
Observer Design Pattern		
Source Observer applications	4/7/20	5/1/20
Test both applications without JIT	4/29/20	5/1/20
Use statistical tests to interpret data	5/3/20	5/3/20



Energy Consumption Of Design Patterns

Anna Davison

Project Start:	Wed, 1/29/2020
Today:	Sun, 5/3/2020
Display Week:	8

TASK	START	END
Strategy Design Pattern		
Source Strategy applications	1/29/20	2/12/20
Test both applications with JIT	2/12/20	2/26/20
Use statistical tests to interpret data	3/2/20	3/3/20
Test both applications without JIT	3/17/20	3/18/20
Use statistical tests to interpret data	3/20/20	3/20/20
Template Design Pattern		
Source Template applications	4/7/20	4/8/20
Test both applications without JIT	4/26/20	4/30/20
Use statistical tests to interpret data	5/3/20	5/3/20
Observer Design Pattern		
Source Observer applications	4/7/20	5/1/20
Test both applications without JIT	4/29/20	5/1/20
Use statistical tests to interpret data	5/3/20	5/3/20

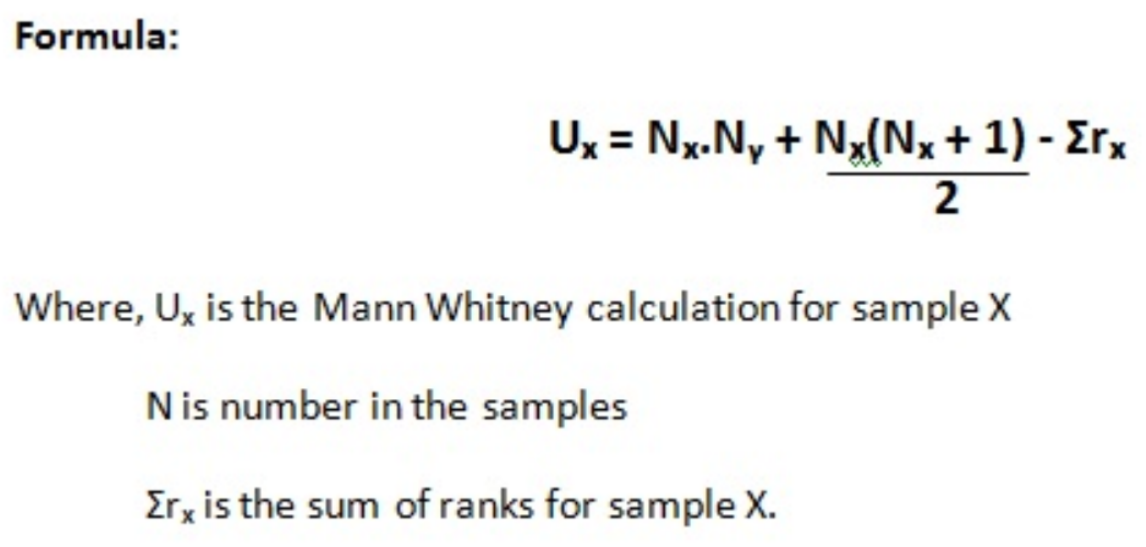
Gantt Chart of how the project actually proceeded

Chapter 5: Statistical Planning

This chapter introduces the statistical tests that were performed on the data collected.

5.1 Statistical Tests

In anticipation that the data would not necessarily be normally distributed, it was decided to generate a range of Summary Statistics with the data, utilising the Data Analysis Toolpack in Microsoft Excel. It was further decided to run the Mann-Whitney U test: This is a non-parametric test that compares two groups without assuming they are normally distributed. [18] An online Mann-Whitney calculator was used to ensure that there were no errors in the calculations. [19] The null hypothesis for this test states that the median of the two samples is the same, as discussed in section 3.3.



Formula:

$$U_x = N_x.N_y + \frac{N_x(N_x + 1)}{2} - \Sigma r_x$$

Where, U_x is the Mann Whitney calculation for sample X

N is number in the samples

Σr_x is the sum of ranks for sample X.

Fig.6. Details on how to perform a Mann-Whitney U test. [20]

5.2 Evaluating the Results

The results were tested with alpha = 0.05. This means that there is a 5% risk of concluding that the null hypothesis is rejected when it should not be rejected.

Chapter 6: Key Learnings

This chapter introduces the components of the project that were considered, changed or dismissed during the conducted experimentation. Section 6.1 discusses the JIT compiler and how it impacted the project. Section 6.2 addresses miscellaneous aspects of the project that were altered and considered and the reasoning behind these decisions.

6.1 Just In Time Compilation

The JIT compiler is a component of the Java Runtime Environment and it is responsible for performance optimisation during run time. [21] As a result of the perceived impact of the performance optimisation, it was decided to measure the energy consumption of the Strategy Design Pattern twice, once with JIT and once without. Based on the results detailed in chapter seven it was decided to complete the remainder of the tests with JIT disabled. There was a significant difference between these results, as shown in the table below. It was determined that it was more appropriate to keep the JIT disabled as it would eliminate a potential cause of variance within the experimental results. The command that was used to run the program with JIT disabled was "java -Djava.compiler=NONE RunTest".

	U	Z-Score	P-Value	Reject Null Hypothesis
Non-Strategy	201	-3.67393	.00024	Yes

Mann-Whitney U test performed on data from JIT and Non-JIT non-Strategy data

6.2 Other Project Considerations

Due to the nature of this project and how small the programs that were being run are in length it was decided that in order to capture the entire duration of the test that each time an application is being tested the program is run 10,000 times. This enabled the testing procedure to generate more accurate results.

Each application was sourced from www.tutorialspoint.com. It was decided to use this site because it had an example program for each design pattern and the software for this project needed to be as unbiased as possible. As a result of this, there may be some bias from the depatterned applications as the author had to generate these. To do this the author went about removing the design pattern from each of the applications. This process varied from application to application but essentially the author worked out what made each application follow a design pattern and then the author refactored it so that it no longer followed this pattern.

In the case of the Strategy design pattern, the strategy and context classes were removed. This

ensured that a specified algorithm was executed at run time.
In the case of the Template Method design pattern, the abstract game class was removed. The football and cricket classes were adapted so that they no longer extended the game class. In the main class, the football and cricket classes were initialised and had their methods called separately. This removed the Template Method design pattern as they were no longer following the template laid out in the game class.
In the case of the Observer design pattern, the observer and subject classes were removed. As a result, each time an object that used to be an observer needed to have its state updated, this needed to be performed manually and individually. Before the application had the design pattern removed, the state was updated automatically for each observer whenever the universal state was updated.

It had previously been decided to test each design pattern with three applications. As the project progressed the decision was made to only test each design pattern with one application as it was indicated that additional testing would not provide any additional corroborating or contradictory information and there was insufficient time to complete the necessary tests.

It was suggested that the energy consumption of the computer should be measured while running in safe mode. This was attempted however the computer was unable to detect the meter while it was in safe mode so it was decided that this was an unnecessary element to add to the experiment. Instead the internet was disconnected on the laptop and applications and processes that were running were disabled before beginning to record the energy consumption.

Chapter 7: **Evaluation**

This chapter assesses the results from the statistical tests. Section 7.1 includes the URL for the Github repository where the software, data and some statistical tables are stored. Section 7.2 presents the summary statistics generated from the data collected. Section 7.3 reviews the results from the Mann-Whitney U tests.

7.1 Results

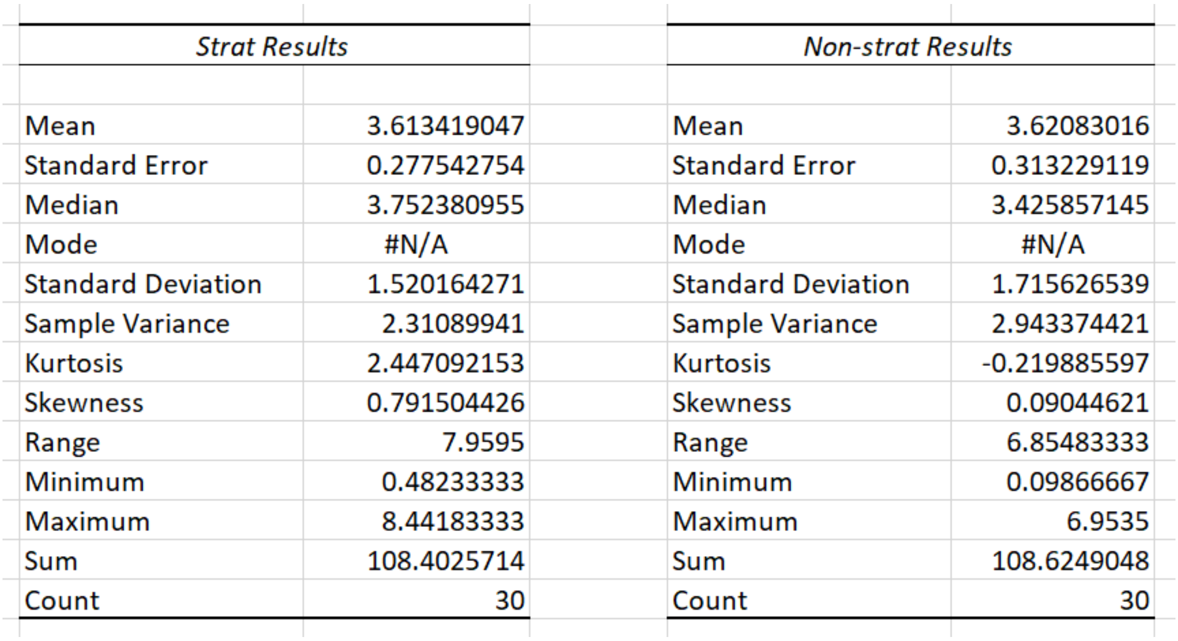
All data and summary statistical tables can be found on Github.com through this URL:

https://github.com/Davison129/EnergyConsumptionOfDesignPatterns.git

7.2 Summary Statistics

The summary statistics from the data collected are as follows.

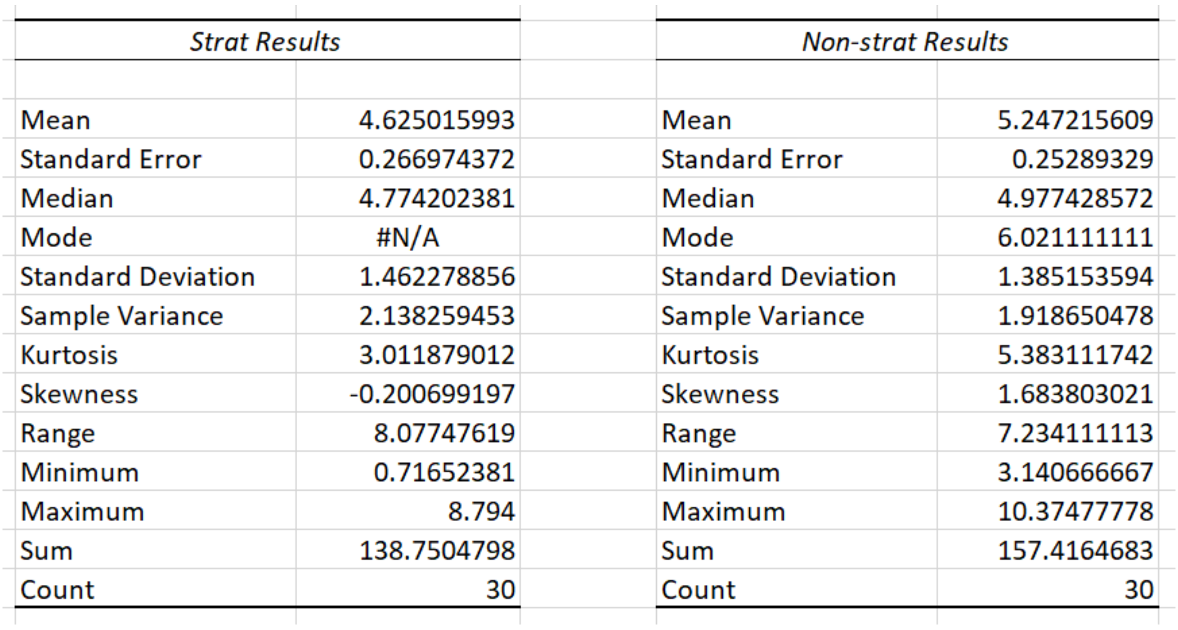
Strategy Application:



Strat Results	
Mean	3.613419047
Standard Error	0.277542754
Median	3.752380955
Mode	#N/A
Standard Deviation	1.520164271
Sample Variance	2.31089941
Kurtosis	2.447092153
Skewness	0.791504426
Range	7.9595
Minimum	0.48233333
Maximum	8.44183333
Sum	108.4025714
Count	30

Non-strat Results	
Mean	3.62083016
Standard Error	0.313229119
Median	3.425857145
Mode	#N/A
Standard Deviation	1.715626539
Sample Variance	2.943374421
Kurtosis	-0.219885597
Skewness	0.09044621
Range	6.85483333
Minimum	0.09866667
Maximum	6.9535
Sum	108.6249048
Count	30

Fig.7. Summary Statistics for JIT enabled Strategy Application.

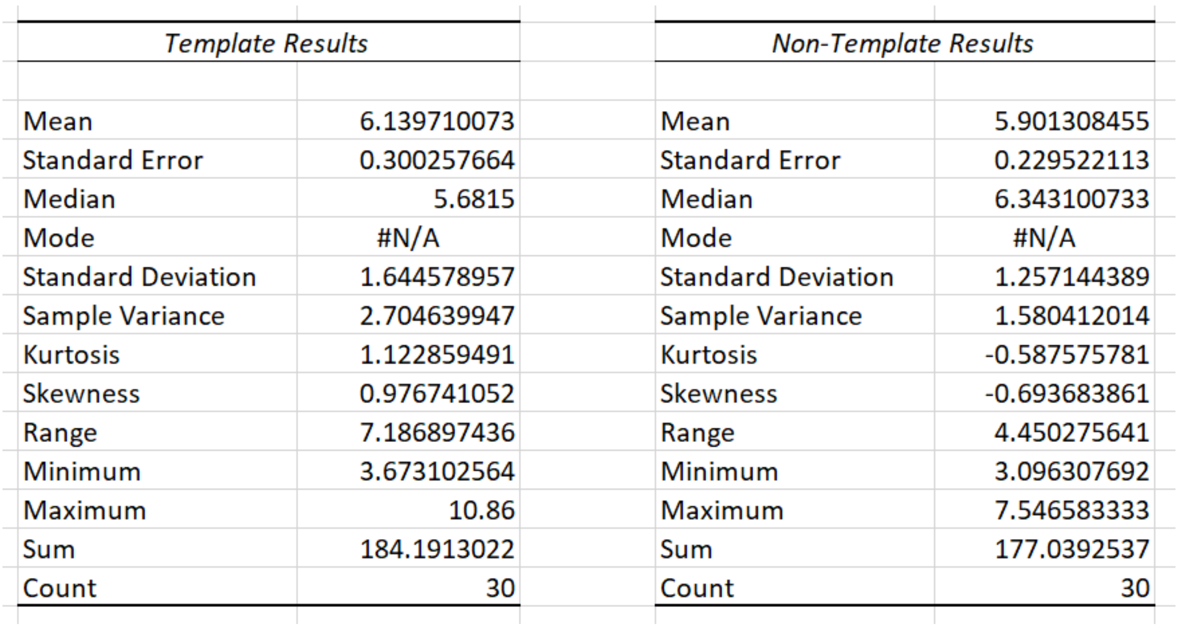


Strat Results	
Mean	4.625015993
Standard Error	0.266974372
Median	4.774202381
Mode	#N/A
Standard Deviation	1.462278856
Sample Variance	2.138259453
Kurtosis	3.011879012
Skewness	-0.200699197
Range	8.07747619
Minimum	0.71652381
Maximum	8.794
Sum	138.7504798
Count	30

Non-strat Results	
Mean	5.247215609
Standard Error	0.25289329
Median	4.977428572
Mode	6.021111111
Standard Deviation	1.385153594
Sample Variance	1.918650478
Kurtosis	5.383111742
Skewness	1.683803021
Range	7.234111113
Minimum	3.140666667
Maximum	10.37477778
Sum	157.4164683
Count	30

Fig.8. Summary Statistics for JIT disabled Strategy Application.

Template Method Application:

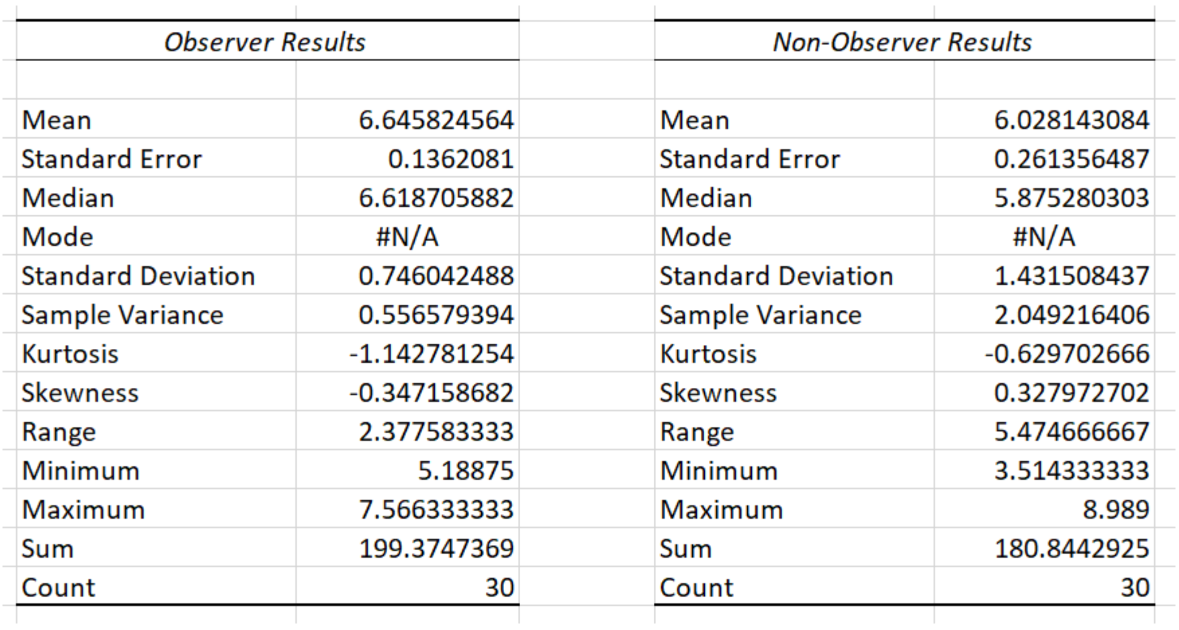


Template Results	
Mean	6.139710073
Standard Error	0.300257664
Median	5.6815
Mode	#N/A
Standard Deviation	1.644578957
Sample Variance	2.704639947
Kurtosis	1.122859491
Skewness	0.976741052
Range	7.186897436
Minimum	3.673102564
Maximum	10.86
Sum	184.1913022
Count	30

Non-Template Results	
Mean	5.901308455
Standard Error	0.229522113
Median	6.343100733
Mode	#N/A
Standard Deviation	1.257144389
Sample Variance	1.580412014
Kurtosis	-0.587575781
Skewness	-0.693683861
Range	4.450275641
Minimum	3.096307692
Maximum	7.546583333
Sum	177.0392537
Count	30

Fig.9. Summary Statistics for JIT disabled Template Method Application.

Observer Application:



Observer Results	
Mean	6.645824564
Standard Error	0.1362081
Median	6.618705882
Mode	#N/A
Standard Deviation	0.746042488
Sample Variance	0.556579394
Kurtosis	-1.142781254
Skewness	-0.347158682
Range	2.377583333
Minimum	5.18875
Maximum	7.566333333
Sum	199.3747369
Count	30

Non-Observer Results	
Mean	6.028143084
Standard Error	0.261356487
Median	5.875280303
Mode	#N/A
Standard Deviation	1.431508437
Sample Variance	2.049216406
Kurtosis	-0.629702666
Skewness	0.327972702
Range	5.474666667
Minimum	3.514333333
Maximum	8.989
Sum	180.8442925
Count	30

Fig.10. Summary Statistics for JIT disabled Observer Application.

The results from the Strategy application appear to be not-normally distributed as shown in Fig.7 and Fig.8 above. This is shown by the fact that the kurtosis value and the skewness value are not within the range of minus two to plus two (kurtosis measures the sharpness of the peak in a frequency curve while skewness measures how symmetrical the data collected is). [22] In comparison, Fig.9 and Fig.10 appear to be normally distributed as their kurtosis and skewness are within this range.

The mean energy consumption for the depatterned applications is higher than the design patterned applications for the Template Method and Observer design pattern applications. The reverse is true for both methods of testing of the Strategy design pattern application.

The minimum and maximum values from Fig.7 to Fig.10 do not appear to follow any particular pattern and can be discounted. The summed values from Fig.7 to Fig.10 appear to follow the same pattern as the mean energy consumption. This is logically correct as the mean is calculated by dividing the sum by the sample size and this remains consistent from Fig.7 to Fig.10, so it is understandable that the sum follows this pattern also. The median (when the number of values being evaluated is an even number) is the mean of the two values in the middle of the list when the list is ordered. The median values do not follow the same pattern as mean energy consumption, due to the variance in outliers from application to application tested.

7.3 Mann-Whitney U Test

The results for the Mann-Whitney U Test(.05 significance and two-tailed) are:

	U	Z-Score	P-Value	Reject Null Hypothesis
JIT enabled Strategy	448	-0.02218	.98404	No
JIT disabled Strategy	344	-1.55976	.11876	No
JIT disabled Template Method	441.5	0.11828	.90448	No
JIT disabled Observer	317	1.95894	.05	No

As demonstrated in the table above, the null hypothesis was not rejected for any of these cases. This means that this test does not detect a significant difference between any of these sample groups. However, as demonstrated in Fig. 11 above, both the mean and the median for the Observer application show significant increases in energy consumption when the design pattern is utilised. Despite the fact that the Mann-Whitney U test did not discover a statistically significant difference, there is clearly little difference between the JIT enabled Strategy applications with and without design pattern. With JIT disabled, the depatterned Strategy application consumes more energy. With JIT disabled, the design patterned Template Method and Observer applications consume more energy.

Chapter 8: Summary and Conclusions

8.1 Summary

This report contains the plan to test the hypothesis as well as the actual method used to test the hypothesis. Applications to be tested were gathered from external sources. The energy consumption of the applications was recorded both with and without design patterns. Statistical tests were performed on the results and conclusions were drawn from the results of the tests. The difference in energy consumption of all the applications were not considered significant by the Mann-Whitney U test.
The Strategy application showed little difference in energy consumption and the JIT was enabled.
The Strategy application showed a decrease in energy consumption when the design pattern was used and the JIT is disabled.
The Template Method application showed an increase in energy consumption when the design pattern is used and the JIT is disabled.
The Observer application showed a larger increase in energy consumption when the design pattern is used and the JIT is disabled.

8.2 Conclusions

Through the experiments that were performed in this project, no significant difference was detected between the Design Patterned applications and the equivalent depatterned applications. The results overall were inconclusive: Design Patterns may have an impact on energy consumption, however, more research is required in this topic to investigate the energy consumption of a wider variety of Design Patterns to obtain a clearer understanding of this area.

Acknowledgements

I would like to thank my supervisor, Dr Mel Ó Cinnéide, for his advice and support throughout this project. I would also like to thank my father, Bobby Davison, for proof-reading this report. I would like to express my appreciation to Deaglan Connolly Bree, a PhD student at UCD, for his support in this project.

Bibliography

1. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.
2. Joshi, R. *Java Design Patterns* (Java Code Geeks, 2015).
3. https://data.worldbank.org/indicator/eg.use.comm.fo.zs.
4. https://www.rte.ie/news/2019/1108/1089500-esb-power-plants/.
5. https://www.eia.gov/todayinenergy/detail.php?id=41433.
6. https://www.rte.ie/news/ireland/2019/0325/1038381-energy-electric-bruton-2030/.
7. https://www.britannica.com/science/energy.
8. https://www.solarschools.net/knowledge-bank/energy/types.
9. Litke, Zotos, Chatigeorgiou & Stephanides. *Energy Consumption Analysis of Design Patterns* (Proceedings of World Academy of Science, Engineering and Technology, vol.6, 2005).
10. Bunse & Stiemer. *On the Energy Consumption of Design Patterns* (Softwaretechnik-Trends, 2013).
11. Jäger, Isabella, Smets, van Swaaij & Zeman. *Solar Energy, Fundamentals, Technology and Systems* (Delft University of Technology, 2014).
12. https://www.britannica.com/science/newton-unit-of-measurement.
13. Kerrison, Buschhoff, Nunez-Yanez & Eder. *Measuring Energy* (ICT - Energy Concepts for Energy Efficiency and Sustainability, 2017).
14. https://cacm.acm.org/magazines/2011/5/107684-an-interview-with-steve-furber/fulltext.
15. Hirst, Miller, Kaplan & Reed. *Watts Up? Pro AC Power Meter for Automated Energy Recording, A Product Review* (Behavior Analysis in Practice vol.6, 82-95, 2013).
16. https://arcb.csc.ncsu.edu/~mueller/cluster/arc/wattsup/metertools-1.0.0/docs/meters/wattsup/manual.pdf.
17. https://www.fishersci.com/shop/products/watt-s-up-power-meter-watts-up-pro/s66185.
18. https://www.socscistatistics.com/tests/mannwhitney/.
19. https://www.socscistatistics.com/tests/mannwhitney/default2.aspx.
20. https://www.mbaskool.com/business-concepts/statistics/8580-mann-whitney-u-test.html.
21. https://www.geeksforgeeks.org/just-in-time-compiler/.
22. https://youtu.be/EG8AF2B_dps.