AutoRSpec

Dan Shreeve

2017-04-25

Signed declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Daniel Demaine Shreeve
Signature: .....
Date: 03/05/2017

Abstract

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Acknowledgements

First and foremost I would like to thank my supervisor **Dr. Gordon Fraser** for accepting my proposed project and providing impeccable advice and guidance throughout the process.

Finally I would like to thank my parents, **Linda Shreeve** and **Paul Shreeve**, and my grandparents **Elsie Marsden** and **Ray Marsden** for giving me the oppurtunity to go to university and supporting me throughout.

Contents

1 Chapter 1: Introduction

1.1 Importance of testing

How severe can the consequences be from an error in a piece of software? In 1983 a bug in a piece of software nearly started world war three.

During the Cold War, 1983, tensions between the US and Soviet Russia were extremely high. A Soviet early warning system had detected the launch of 5 ballistic missiles from the US. The only reason that Russai did not retaliate, starting World War Three was the fact that Lt Col Stanislav Petrov had a "...funny feeling in my gut" and concluded that if the US was launching a full scale attack they would launch more than 5 missiles. The error in the software was discovered to be a bug in the part of the software that distinguished false missiles from satellites picking up the reflection of sunlight from the top of clouds.[1] If the bug in the code has falsey detected more missiles the world could be a very differn't place.

The European Space Agency spent 10 years and $7 billion designing and constructing the Ariane 5, a rocket that can launch multiple satellites into orbit from a single launch. 39 seconds into its maiden voyage it exploded, destroying the Ariane 5 and its contents of 4 uninsured, extremley expensive scientific sattelites. The explosion was caused by its own self-destruction sequence which was triggered automatically as the boosters were being torn away by aerodynamic forces. The extreme aerodynamic forces were caused by the rocket trying to recorrect its course due to flight data provided the guidance system. The guidance system had crashed and shutdown, along with its backup, the flight data provided that caused the rocket to readjust its course was actually a diagnostic error message. The cause of the shutdown was the guidance system trying to convert the sideways velocity of the rocket from 64-bit to 16-bit where an overflow occured causing the system to shutdown. To make matters worse, the programmers were aware that it could overflow but that particular variable would never be large enough as it was used to prepare for launch and not in flight. However it was decided the system should run into the first 40 seconds of flight incase of breif hold in launch countdown to make restarting the system easier. A known flaw in a system, that could of been handled ended up causing the chain of events that led to the rocket exploding.[2]

Software disasters are caused by three main reasons. Poor software project management, poor risk assessment and poor development and testing practices.[3] Software testing is therefor extremely prevlent and has upmost importance as it can prevent World War and save companies alot of money.

1.2 Benefits of testing and automation

Inadequate software testing infastructure was estimated to cost the US economy $59.5 billion a year.[4] It was also estimated that the potentail cost reduction from feasbile infastructure would be $22.2 billion a year. [4] Due to software disasters and the vast amount of money that can be saved and also avoid incur-

ring people have become more aware of the importance of testing. The benefits that this saving comes from is the increased reliabilty and quality of the product produced when software testing is implemented.

"In a typical programming project approximately 50 percent of the elapsed time and more than 50 percent of the total cost were expended in testing the program or system being developed"[5]. Software testing saves you alot of money but also costs alot of money. By automating part or the whole process the costs can be reduced while still obtaining all of the benefits.

1.3 What is AutoRSpec

AutoRSpec is a piece of software that will automatically generate RSpec test cases that test model validation in Ruby on Rails Applications. The User will specify there tables and assign properties to fields, maximum length of 5, using a web interface. Once a table is defined the user will be able to generate and download a file that will test all the properties defined using randomly generated values.

The aims of the project are to:

1. Reduce the amount of time it takes for a User to produce tests for model validation
2. Produce tests that are of high readable quality
3. Produce tests that fully test properties specified
4. The process should be hassle free

Challenges that the project faces are

1. Identifying the minimum information required to produce tests
2. Creating a process that is hassle free
3. Natural language in tests that is appears human written
4. Generating the tests in an acceptable time frame
5. Building an efficent database structure for the information

2 Chapter 2: Literature Review and Research

2.1 Tool to use

The main requirements for choosing what tools to use to build the application are:

1. Allowing a User to enter information onto a database
2. Allowing a User to view information on the database

3. A system that can process the information and generate Test cases

An MVC web application fits all these criteria while providing additional benefits. MVC, Model-View-Controller is an architectural pattern that seperates an application into three interconnected parts. The separation allows for responsibilities to be allocated independently to each component, seperating the logic from the user. The model is responsible for the data of the application and the rules and logic used to create and update the information. The view is responsible for displaying the data and possible interactions with the system to the user. The controller is responsible for controlling the flow of the system, accepting user input and converting it into commands for the model and view.

An example of the components interacting would be creating an entry to a database. The view would be responsible for displaying the form in which to fill in. On submission the controller will process the information, ensure only persmissable information is submitted and enter additional information, then send it to the model. The model would verify the structure of the information, correct fields are present and cohere with its rules. The model will then notify the controller if the submission was successful or not and the controller will update the view to reflect the status.

The seperation means that all user interaction with the database has to be verified by the controller. This provides a high level of security as each action is controlled and the internal structure and representation of the information within the database is hidden. Simultaneous development is also possible due to the seperation of the components, work on the front and back end concurrently. Although I will not be able to get the full benefit of this as I am developing the project solo, it will allow me to shift focus as components do not need to finished before switching to another, giving greater flexibility in development.

High levels of cohesion are inherited automatically from the architecture with the grouping of logically similar elements, this makes the code easier to read and creates a more natural flow within the source code. There are however some drawbacks to MVC architecture, they are inheritly more complex due to the seperation and the framework must be learnt in addition to the programming language it is in, which there can be multiple languages between the components. This steep learning curve could mean a large initail investment into a team learning a new framework along with its languages.

MVC web application frameworks have become extremely popular and are behind some of the most used and powerful websites. Django an MTV, follows MVC architecture but its creators decided to rename the components [6] to better suit them, is behind the two most visted websites in the world Google and YouTube[7][8]. Ruby on Rails another MVC is behind Twitter, Airbnb and Soundcloud.[9] MVC frameworks are known for their scalablity, being suitable for the smallest to the largest projects. However FaceBook decided that its scalablity had reached its limit, that adding new features made the code exponentailly more complex.[10] My project will be no where near the scale of facebooks sourcecode so I do not need to worry about reaching the end of its scalibilty.

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Figure 1: C++ print Hello world to console

```ruby
puts "hello world"
```

Figure 2: Ruby print Hello world to console

The chosen MVC to construct the project in is Ruby on Rails. I have done previous projects in both Ruby and the Rails framework, also the novelty of contstructing the software in the software that its output will test.

Ruby was selected as the primary programming language by default as it is the language that runs Ruby On Rails. Ruby is a dynamic, multi-paradigm programming language. The paradigms consist of Object-oriented, Imperative, Functional and Reflective making it a very powerful and versatile language. This combination is from its founder Yukihiro Matsumoto who was influenced by Perl, Smalltalk, Eiffel, Ada, and Lisp.

Rubys primary design goal was to make a language that he himself enjoyed using, by minimizing programmer work and possible confusion(Ruby Wiki). Achieved with a focus on human interaction, how programmers code and design applications as opposed to focusing on how the code will run on machines. And also following the principle of least astonishment, where the behaviour of the language minimizes confusion for experienced users.

The above two images show C++ 1 and Ruby 2 printing Hello world to the console. The comparison between the two languages highlights the efficiency and simplicity of the Ruby language.Ruby on Rails projects are commonly worked on by a group of people and in multiple languages, therefore the simplistic syntax gives greater clarity and understandability to programmers who are lesser

experienced in Ruby.[11]

Ruby is open source, free and redistributable with a vast range of existing code from both Ruby and its large community. Primarily consisting of Gems, code packages that can be installed and supported into a project easily and with minimal effort via RubyGems, and frameworks, such as Ruby on Rails. Making it very popular for education and business. Following the DRY Don't Repeat Yourself principle in a very effective and efficient manner.[11]

Ruby was ranked ninth on TIOBE index[**?**] and has become a very popular and respected language relative to its age among the other languages on the index. No alternatives could be considered due to the dependency of Ruby on Rails on Ruby, however Ruby is a very strong and durable language so it does not detract from the overall project.

2.2 Testing and Automation

2.3 Testing in Rails

2.4 Evaluation

3 Chapter 3: Requirements and Analysis

Filler text, filler text, filler text, filler text, filler text, filler text, filler text, filler text.

4 Chapter 4: Design

Filler text, filler text, filler text, filler text, filler text, filler text, filler text, filler text.

5 Chapter 5: Implemention and Design

Filler text, filler text, filler text, filler text, filler text, filler text, filler text, filler text.

6 Chapter 6: Results and Discussion

Filler text, filler text, filler text, filler text, filler text, filler text, filler text, filler text.

7 Chapter 7: Conclusions

Filler text, filler text, filler text, filler text, filler text, filler text, filler text, filler text. Hello World!

```
for full_iteration in 0..50
  numbers = gen_rand_int_array(500000)
  wanted = numbers.delete_if{ |n| n.method(isolated[0]).(isolated[1])}
  if wanted.empty?
    next
  elsif rest.empty?
    return wanted.sample
  else
    rest.each do |rule|
      wanted.keep_if{ |n| n.method(rule[0]).(rule[1])}
      if wanted.empty?
        break
      end
    end
    if wanted.empty?
      next
    else
      return wanted.sample
    end
  end
end

return false
```

Figure 3: caption for image, shown below image

7.1 subsection

Structuring a document is easy!! [12]

7.1.1 subsubsection

p1 It's a me, Mario[1].

sp1 Wwoohooo

8 image

Figure 3 shows some savage code

References

[1] C. Barker, "The top 10 it disasters of all time," 2007.

[2] J. Gliek, "A bug and a crash," 1996.

[1] [12]

Table 1: Caption for the table.

Some	actual	content
prettifies	the	content
as	well	as
using	the	booktabs package

[3] P. A. McQuaid, "Software disastersunderstanding the past, to improve the future," *Journal of Software: Evolution and Process*, vol. 24, no. 5, pp. 459–470, 2012.

[4] RTI, "The economic impacts of inadequate infrastructure for software testing," in *Planning Report 02-3 The Economic Impacts of Inadequate Infrastructure for Software Testing*, p. 309, NIST, 2002.

[5] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 2011.

[6] Django, "Django faq," 2017.

[7] SHUUP, "25 of the most popular python and django websites," 2015.

[8] Alexa, "The top 500 websites," 2017.

[9] Coderfactory, "Top 15 sites built with ruby on rails," 2017.

[10] Infoq, "Facebook: Mvc does not scale, use flux instead," 2014.

[11] Ruby, "About ruby," 2016.

[12] J. P. Near and D. Jackson, "Rubicon: bounded verification of web applications," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, p. 60, ACM, 2012.