Improving Quality Team Coding Skills with Code Clubs

Authors: Dwayne Thomas, Kevin Swallow

Email addresses: dthomas@crowdcompass.com, kswallow@crowdcompass.com

Abstract

As CrowdCompass has matured its agile processes, pressure has increased on the company's testing team to become more efficient at delivering high quality products. Testers actively sought coding knowledge to contribute more to software development discussions. Testers also sought to automate more of the testing workflow. The need for the code discussions arose because very few of the testers were experts with programming. Moreover, most of us had less than 2 years of experience at CrowdCompass and did not know how else to speed our adoption of coding best practices.

Code Clubs are recurring coding discussions that often feel like a brewing stew of ideas. They give us the nutrients to strengthen our strategic effort of leveraging code in our testing efforts. CrowdCompass leadership wholly encourages the clubs, because learning and initiative are company values --and that is reflected in our quality engineering (QE) practices. We believe the organization has benefited from these low investment meetings. No direct causal link can be established between product improvements and the clubs, but, since the CrowdCompass Code Clubs started, many of the test staff have started new scripting projects and even some of our most code-anxious testers have contributed to the automation effort.

This paper shares the experiences of our testing team with these code discussions. We will provide the information necessary for the reader to start their own Code Club and some of the programming lessons we learned.

Biography

Dwayne Thomas is a quality engineer for CrowdCompass by Cvent. He's mainly tested in the television/software space, before coming into mobile conference software. Dwayne caught the learning bug when he taught middle school math for 4 years in New York City.

Kevin Swallow was an Oregon licensed professional mechanical engineer before moving into software quality and automation. He graduated magnum cum laude from CSU-Sacramento and is currently working at CrowdCompass, where he promotes software best practices and helps to implement automated testing.

1. Introduction

As Crowdcompass has matured its agile processes, pressure has increased on the company's testing team to become more efficient at delivering high quality products. Testers actively sought coding knowledge to contribute more to software development discussions. Testers also sought to automate more of the testing workflow. The need for the code discussions arose because very few of the testers were experts with programming. Moreover, most of us had less than 2 years of experience at CrowdCompass and did not know how else to speed our adoption of coding best practices.

Code Clubs are recurring coding discussions that often feel like a brewing stew of ideas. They give us the nutrients to strengthen our strategic effort of leveraging code in our testing efforts. . CrowdCompass leadership wholly encourages the clubs, because learning and initiative are company values --and that is reflected in our quality engineering (QE) practices. We believe the organization has benefited from these low investment meetings. No direct causal link can be established between product improvements and the clubs, however, since the CrowdCompass Code Clubs started, many of the test staff have started new scripting projects and even some of our most code-anxious testers have contributed to the automation effort.

This paper shares the experiences of our testing team with these code discussions. We will provide the information necessary for the reader to start their own Code Club. As a storytelling device, we at times compare process of starting a code club to cooking processes. We include pieces of code in 'single' quotation marks in order to make audiences familiar with the content of our discussions. Electronic links are underlined throughout the paper and listed fully in the references section at the end of the paper.

2 The CrowdCompass Code Club

2.1 History

The idea for the CrowdCompass Code Club started simmering after Dwayne Thomas determined that a Code Club was an excellent opportunity to fill the QE team's skills gaps. QE manager Lloyd Bell enthusiastically supported the idea. Most of the testing team joined in. As news of our efforts wafted out to the rest of our organization, some developers started contributing to it as well.

2.2 How to start your own Code Club

Start your own Code Club to start experimenting with the benefits of it. The following paper is a collection of ideas highly relevant within the CrowdCompass context. However, everyone in your organization could join the Code Club. Simply put, the clubs are informal gatherings of 3-6 testers discussing 100 lines of code for one hour. We call our club "Codetry." Sessions proceed by testers questioning any new terms or practices in the scripts. Internet searches help explain new terms. An experienced software developer is vital to understanding best practices.

We scheduled meetings in Microsoft Outlook and kept them at about 85% regularity. We created an ongoing instant message room to capture fleeting, but essential, learning points. The notes helped folks that were not able to attend the meeting. We had varying success using Google Docs for planning ideas for future sessions. We found the meetings were easiest to organize on a week by week basis, since we wanted to make the meetings highly relevant for those who could attend.

We also think it is important to share efforts early and often to the rest of your organization. Sharing helps to recruit new members and get new resources.

2.3.1 Early Reading Code Good recommendations

Our Code Club draws on many sources--starting with the foundation of offerings on www.readingcodegood.com. These sources were great for beginning since they were readable and had

engaging applications. The Python and Ruby programming language scripts often met our criteria and we were quickly able to look at the behavior they implemented. Nevertheless, we tried to be inclusive of many languages in our meetings.

For example, the Google Books repository helped us understand how to pull information from the Google Books application programming interface (API). The interfaces enable the exchange of information among systems that use different technologies. They can skip the presentation layer of data transfer. Since CrowdCompass has branches on the East Coast and we share their API's, API functions had enough relevance to us testers. For example, API discussions broached how the Ruby language includes external libraries (it uses the require keyword). Other folks at the meeting offered that Python uses the 'import' keyword to accomplish the same task.

Another Readingcodegood recommendation, the Country and Region select script, showed us how to programmatically translate country names and introduced us to internationalization and UTF-8 character concerns (Sstephenson, 2014). Our testing group decided pieces of code snippets like this Joel on Software (2003) one could be relevant if and when our company expands its applications to non-US markets. As states the article states we certainly would want our application pages to translate as more than a string of question marks.

2.3.2 Keeping it simple, but not too simple

When we started the Code Club effort, our goal was to keep things simple and comprehensible. At first we avoided file-spanning object oriented concepts such as inheritance, encapsulation, polymorphism, and abstraction. These concepts had especially nonlinear logic for the tester group. Also these concepts might have hidden code implementation from testers. In later meetings we tackled these topics (see appendix).

2.3.3 Seeking breadth

We needed familiarity with finding code on GitHub to expand the topics to review. Github is an online community where developers share more than 24 million code projects. This search finds code repositories of less than 100 kilobytes (Github, 2015). Since code files are text based, this search for small files proved more than adequate for the first few sessions of Code Club. Participants tended towards ruby and python repositories, the languages of our testing harness and our company's applications. Files with the Ruby (.rb) and Python (.py) extensions were most fruitful for us: they contained good working code examples. Python code is so readable, for example, that this code will loop and print the 3 strings that the reader might expect it to:

```
'for x in my_range(1, 10, 0.5):     print x'
```

Still, we found that different languages had enough idiosyncrasies that we could shop among them and learn their conveniences and handicaps.

Github also introduced us to project file structure on the internet. We found that lib directories (folders named lib) often contained scripts that were appropriate for our review. We learned to ignore other files, such as vagrant files (.vag), as they seem to only contain lower level information that was more understandable to computers than people. We still have to revisit this kinds of files to learn more about them.

2.3.3 Discussing the Python script implementations

At Code Club we asked questions about implementations not specific to our business logic. One attendee asked "what does `if __name__ == "__main__":` do?" The online programming question-and-answer forum Stackoverflow helped us answer that before executing the code, Python will define a few special variables. MAIN is an entry point for the language. By doing the main check, the code only executes when needed.

2.3.4 Less perfect scripts

We made a point to learn code from scripts of varying levels of expertise. The Rayel program showed us how Java's print statements ('System.out.println') are more complicated than Python and Ruby (Amanoske, 2015). Java's implementation made sense to us since Python ('print') and Ruby ('puts') are more interactive languages and Java has to be compiled before being used. During these exercises, having a software developer in the group was especially helpful for going through the obtuse code. The developers helped us discuss code quality as well as its structure, syntax, function and implementation.

3 Complex ingredients and flavors

As we learned, we became more confident and knowledgeable. Early scripts led us to more involved ones which could be combined with our company's code stew. We more confidently analyzed some of the more complex code previously mentioned.

3.1 Design patterns

Discussions about scripting, best practice, and application design led to our covering design patterns explicitly. These are frequently-used code structures for object creation and method heuristics. We focused on patterns that were more important for interactive languages such as Python or Ruby and/or relevant to our work. We looked at the Strategy Pattern, the Model View Controller pattern, the Template Method Pattern and decorators (Faif, 2015).

As an example, for mainly his own practice, one server developer at our company concocted a math polynomial formula to explain the template method pattern. He showed that it is possible to reuse the math distance operation with an assortment of polygons, for example from linear to triangles. He used the template method pattern to implement the DRY (do not repeat yourself) mantra. The pattern allows us to change specific steps of methods without changing the method's structure. The same pattern was especially instructive for the testing team's understanding of our testing harness. The operation of logging in and out of an app could similarly be reused. Notably, the template pattern is the key concept in inheritance, which we initially shied away from.

3.1.2 Model view controller pattern

The model view controller, common in applications, was a discussion topic for the testing team. Figure 1 of the pattern shows that between the computer and the browser there are several layers of script that typically deliver the expected user experience. Each layer of the application is separate and can even be written in a different programming language.

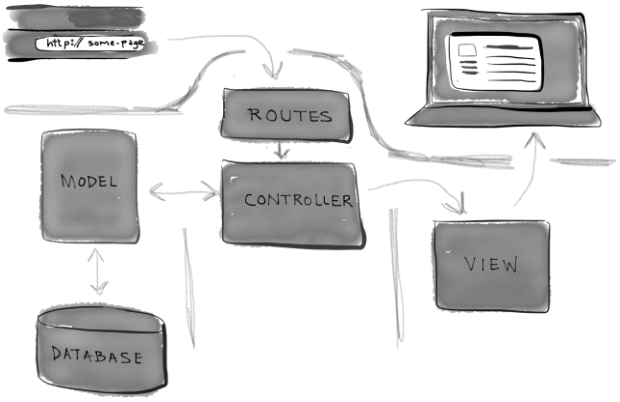

Figure 1: Model view controller diagram

3.2 Configuring servers for deployment

Eventually, Code Club acquainted us with difficulties in deploying servers. Deploying servers is a common practice of members of the CrowdCompass operations team. The INI file format, often found in repositories, is an informal standard for configuration files for some platforms or software. INI files are simple text files with a basic structure composed of sections, properties, and values. In MS-DOS and 16-bit Windows platforms up through Windows ME, the INI file served as the primary mechanism to configure operating system and installed applications features (Wikipedia, 2015). The Ansible script is a radically simple IT automation platform that makes your applications and systems easier to deploy (TRBS, 2015). It avoids writing scripts or custom code to deploy and update your applications— automates in a language that approaches plain English, using text based shell sessions to install on remote systems.

4. Family recipes: learning the company product

Kevin Swallow recognized one of the great benefits of the Code Club was to give the participants insight into our own applications, and motivated them to further study in that direction. Outside of Code Club, testers have installed the company's app onto their local computers under the guidance of developers and preview development features. Testers might not start with the company's app since setting it up requires using development tools.

4.1 Ruby on Rails app walkthrough

4.1.1 Design principle

We began with a Code Club meeting that reviewed one of our Ruby on Rails app repositories. We also referred to the online Install Rails (2015, Griffel) documentation. Attendees learned about convention over configuration. When the convention implemented by the tool matches the desired behavior, it behaves as expected without the developer having to write configuration files. A URL can describe a program entry point, a controller, a method, perhaps an object. Controllers pull data from similarly named models and the output cascades to similarly named views. Only when the desired behavior deviates from the implemented convention is explicit configuration required.

4.1.2 Organization of app

We also learned about the default structure of a rails application. When we see errant behavior in an interface with an https URL, we know to look for config>routes.rb. This file helps us follow web requests into the application and to controllers. This knowledge is sure to help testers identify bugs.

4.2 Tests in Rails

At another session, one of developers demonstrated how unit tests were developed and where they could be found in our application. By understanding the tests that the developers include in the app, we expect to be able to limit our own work to avoid redundancy, and find areas that we might want to include in additional testing. The first step is to find and understand them.

4.3 Debugging

Some of our more focused debugging sessions are still in the works. We have tentative arrangements for developers to tell us how they start with a bug ticket, find the error in the code, and fix it. We hope this knowledge will improve how testers write bug tickets for developers.

4.4 Demoing C and iOS unit tests

Code club provided a venue for programmers and testers to discuss the C language and unit tests. In the course of learning about getters and setters of the C language, we learned about the percentage of the test coverage on our iOS devices. Getters and setters are needed to prevent the need to modify every file in the codebase with variable changes. The also allows for validation in one location and a hidden internal representation elsewhere. For example 'getAddress()' could actually be getting several fields. Getters also prevent setting temporary variable names. This knowledge could also help test engineers learn how to structure their tests so that they make defect isolation efficient.

5 Juggling knives: unexpected tools, new skills

5.1 Jira automated notification of ticket statuses

Code Club trained us to write scripts to facilitate the administration of our software development. Some of these scripts were very exciting and quickly useful. One of these was a script that would automatically execute a Jira query, and then publish a notification to the appropriate chat room. Some testers improved their script to a twice a day notification about Jira tickets that needed to be reviewed by product managers. If product managers do not get to the Jira tickets in a timely manner, testers eventually get blocked on their testing. Therefore our release team devised a script to automatically notify the Product team about potential blockers. The Jira Notifier script is just a few steps away from being a full-fledged bot, which would listen to the Hipchat dialog for commands and execute scripts on a time and calendar basis as well. We'll have a follow-up Code Club session when the first QA Bot comes on line, and it's predicted that each of our sprint teams will have their own.

5.2 The more you know

We reviewed the code for the automated notifier with its author which, in turn, introduced us to the open source python library CLICK(2015). CLICK is the Command Line Interface Creation Kit and is a neat tool for testers who might drown in their own command line creations without appropriate help documentation at their own fingertips This library has become a staple of most of the testing team's scripts.

5.3 Impacts on the testing team and the company

Code Club seems to have had significant impacts. Discussion of coding practices among the quality team members has become more frequent and more informed. At least one quality team member who once adamantly refused to have anything to do with automation is now writing automated tests. Attendance at code-centric brown bags regularly includes a core of 4 attendees and a rotating cast of about 15 participants. In a response to Kevin's suggestions to push the Code Club more solidly toward improving CrowdCompass testing tools, Dwayne installed a local development environment on his computer, and

now several other testers have followed suit. There is talk of establishing local environments for everyone on the testing team.

None of this is happening in a vacuum. One suite of automated API tests is nearly complete, and has had measurable impacts on quality. In fact, it was used to make a major upgrade to one of its endpoints rapidly and with confidence. We help address the testing team's skills gap which allowed us to write more automated tests.There is a definite new sense among all the testers we should code. Developers are also more frequently completing reviews on their own work as well. There is no way to separate which parts of this progression are thanks to Code Club and which parts are simply due to the company zeitgeist.

6 Sumptuous deserts - personal projects

We reviewed some code just for personal projects and plan to do more in the future.

6.1 Webscraping

For life's mundane or stressful moments there might be a script to help things out. The Beautiful soup library (a collection of functions) has proven useful for scraping the web. Web Scraping code potentially helped a tester save money on an Amazon purchase (Shorey, 2015). This Python script uses a government api to search for jobs that do not restrict employment based on criminal records (Shorey, 2015). Another code sends a text when a new Craigslist posting matches a given keyword or phrase (Gjreda, 2015). Individual testers planned to set up a task that runs automatically at timed intervals and saves manual effort.

6.2 Code script for making life easier

Dirmon is a utility which keeps track of changes in a user's current directory (Ayancey, 2015). It is useful for monitoring folders that are not on the web. This script would be useful for telling when to sync folders. The use case for this script might be to regularly update any personal versions of larger shared projects.

Conclusion

Our Code Club originally centered on developing general code fluency for testers. As we narrowed our focus to emphasize code in test, we were glad to see more developers dropping in. We think this is a good thing, and we plan to continue to promote test automation and better testing through knowledge and understanding of coding in general and our company's products in particular. At the same time there is plenty of good material to review outside that scope, and we plan to stray from the narrow path whenever there is a need.

We believe that Code Clubs has been a worthy inclusion in CrowdCompass practice that has benefitted everyone at all levels. For the company, it has been a low-investment training opportunity. Thankfully, many co-workers stepped forward to help us. Understanding code and code principles leads to more involved discussions about code changes. Such discussions likely help prevent bugs from happening or else help bugs to be found sooner. The participants have been provided opportunities for professional development and knowledge sharing among the team and across the company. The authors found that even, with the varied coding experiences, most of the testing team benefited from the discussions in reviewing important practices and concepts. The discussions also confirmed learning programming can be very empowering, allowing users and companies to manipulate an existing product to serve their own interests.

Recommended readings

Hartley, Brody. I Don't Need No Stinking API: Web Scraping For Fun and Profit. https://blog.hartleybrody.com/web-scraping/ (accessed June 1, 2015).

Real Python. "Model-View-Controller (MVC) Explained -- With Legos" https://realpython.com/blog/python/the-model-view-controller-mvc-paradigm-summarized-with-legos/ (accessed March 10, 2015).

Suggested code for review

Amanoske. "Rayel." https://github.com/amanoske/Rayel/blob/master/Rayel.java (accessed Nov 8, 2014).

Ahawker. "Appthwack." https://github.com/appthwack/appthwack-python/blob/master/appthwack/appthwack.py (accessed July 30, 2015)

Ayancey. "Dirmon." https://github.com/ayancey/dirmon (accessed February 23, 2015).

Github.

https://github.com/search?utf8=%E2%9C%93&q=size%3A%3C100+&type=Repositories&ref=searchresults

(accessed May 30, 2015).

Ivanoats. "FizzBuzz." https://github.com/codefellows/FizzBuzz/blob/master/Ruby/fizzbuzz.rb (accessed December 9, 2014).

Faif. "Python-Patterns." https://github.com/faif/python-patterns/blob/master/strategy.py (accessed May 30, 2015).

Shorey, Rachel. "Buscando." https://github.com/aliyarahman/buscando/blob/master/db_populate.py (accessed January 13, 2015).

Shorey, Rachel. "Second_Chance_Employers." https://github.com/aliyarahman/second_chance_employers/blob/master/usajobs.py (accessed January 20, 2015).

Sstephenson. "Country_and_Region_Select." https://github.com/basecamp/country_and_region_select/blob/master/lib/country_select.rb (accessed December 16, 2014).

TRBS. https://github.com/ansible/ansible/blob/devel/examples/scripts/yaml_to_ini.py (accessed January 27, 2015).

Waldherr, Simon. "IRCLogger." https://github.com/SimonWaldherr/ircLogger.go/blob/master/ircLogger.go (accessed February 3, 2015).

Wikipedia. "INI files." http://en.wikipedia.org/wiki/INI_file (accessed July 30, 2015).

Zeantsoi. "GoogleBooks." https://github.com/zeantsoi/GoogleBooks (accessed December 2, 2014).

References

Griffel, M. “Install Rails.” http://installrails.com/ (accessed May 12, 2015).

Richardson, Leonard. Beautiful Soup Documentation.

http://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed June 1, 2015).

Ronacher, Armin. $Click. http://click.pocoo.org/4/ (accessed June 1, 2015).

Stackoverflow. “What does if name do.” http://stackoverflow.com/questions/419163/what-does-if-name-main-do (accessed June 1, 2015).Yitbarek, Saron. “Reading Code Good.” http://www.readingcodegood.com/ (accessed June 1, 2015).

Glossary

Inheritance is when classes in the lower hierarchy get variables (static attributes) and/or methods (dynamic behaviors) from the higher hierarchies.

Encapsulation is a protective barrier that prevents the code and data being randomly accessed by other code defined outside the class. See figure 2 where organs and systems of the cat have been abstracted away.

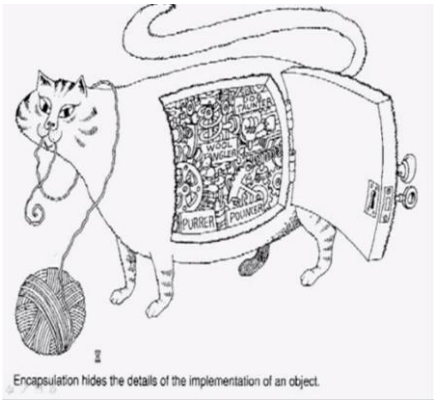

Figure 2: Encapsulation diagram that hides the systems of the cat

Polymorphism takes advantage of inheritance and allows objects to behave differently in different contexts, for example the + (plus) operator in C++: provided integers,

for example, 4 + 5, it performs integer addition to produce 9. Provided with floating point numbers like 3.14 + 2.0, it performs floating point addition, which is a very different operation for a computer, and returns 5.14. Finally given string values, s1 + "bar" it will do string concatenation and return foobar (assuming s1 = “foo”)!

Abstraction is what software developers use to decompose complex systems into simpler components. For example, imagine a video game where we have chosen to abstract a person to their height and their ability to run as a class.

Appendix

Abstraction example: Java code that makes the person run as a function of height, ignoring the other complications of personhood.

```
Public Class Person
   Private _height As Int16
   Public Property Height() As Int16
      Get
         Return _height
      End Get
      Set(ByVal Value As Int16)
         _height = Value
      End Set
   End Property
 Public Sub Run()
   End Sub

End Class
```