Social Coding Standards on TouchDevelop: An Empirical Study

By

Shivangi Kulshrestha

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2017

Acknowledgements

I would like to thank my supervising professor, Dr. Christoph Csallner for

his continuous support, endless patience and trust in my skills. He has been a great mentor always motivating me and rooting for me. His vision and foresight often helped me find new ways to try to solve problems.

I am grateful to Dr. Bahram Khalili and Mr. David Kung for serving as members of thesis committee and for your insightful comments and questions.

Finally, I would like to thank my father, mother and sister for their support and trust in me throughout my life.

Nov 16, 2017

Abstract

Social Coding Standards on TouchDevelop: An Empirical Study

SHIVANGI KULSHRESTHA, MS

The University of Texas at Arlington, 2017

Supervising Professor: Christoph Csallner

This study compares and contrasts the application development pattern on Microsoft's mobile application development platform with leading version control and social coding sites like Github. TouchDevelop is an in-browser editor for developing mobile applications with the main aim to concentrate on 'touch' as the only input. Apart from being the first of it's kind platform, TouchDevelop also allows users to upload their script directly to cloud. This is what makes this study interesting, since the API data of the app has never been studied before to follow social coding standards or version control techniques. Till today, all major IDEs, e.g NetBean, Eclipse etc have plugins to connect to social coding sites like GitHub, BitBucket etc. The same can be said about version control. To upload or sync one's code to cloud, we need a third party software like TeamFoundationServer or SourceTree connected with the physical editor on your machines. TouchDevelop however, let's you directly upload your script to cloud without the help of these tools. This makes it very easy for developers to do a direct version control and follow the social coding standards. This study

facilitates the theory that TouchDevelop can use this particular feature to it's advantage and become one of the leading mobile application development platforms. So far, studies have concentrated on the unique feature of this app, which is using 'touch' as the only input, and, the moving the traditional mobile app development from an editor to directly developing apps on a physical mobile device, like, touch tablets, touch mobile phones. There are no studies which study the version control available in the TouchDevelop IDE, and the ease with which one can access their scripts and scripts of other users. Also, TouchDevelop allows users to comment, like and review scripts, which is in lieu with today's social coding practices. Looking at TouchDevelop from this perspective has not been done yet, and this study concentrates precisely on that.Our study has come up with patterns and trends on the way TouchDevelop apps are implemented and stored. We have studied the relation between comments and reviews and the updates of existing scripts by different users. This has given us empirical proof that TouchDevelop has been operated as a version control tool as well. Our studies are confirming the hypothesis that this editor can be used to directly access and practice social coding protocols without additional third party softwares. If this feature of TouchDevelop is taken advantage of, this IDE becomes one of its kind to directly employ version control without using any plugin or any other third party tool. This can be further extended to apply continuous integration on cloud, which would definitely make TouchDevelop even more alluring, with respect to DevOps.

This helps in TouchDevelop acting as more than just a training tool, and actually being used professionally with increased usage, scripts and projects.

For almost over two decades now, we have increasing presence of social media in our lives. This has given birth to a new trend of collaboration and coding when it comes to software development. A lot of leading products today are open sources and their data is publicly available to manipulate or study.

We know that professionals are working with strict version control systems in place like Github/TFS. This means that nothing gets committed without going through a version control mechanism in the software development cycle. Almost every leading or medium size software firm uses some form of social collaborative platform and does version control through them. There are various new aspects to this style of programming, which has been addressed in this work. As we are moving forward with more diverse software and increasing ease of access via cloud,developers are changing their methods and practices to accommodate the versatile needs of changing development environnments(deployment in cloud). A culture of shared testing on social coding sites and continuous integration of software via the same platforms has become the norm.

This study revolves around mining and observing data from TouchDevelop. TouchDevelop is an app development platform on mobile devices. With an increase in mobile devices becoming the prevalent computing platform, TouchDevelop came together with a simpler solution to making mobile apps as opposed to the traditional practice of first developing an app on an independent editor and then connecting a simulator to test the app. The platform is devised with only touch input in mind and caters to those who want to develop apps using symbols (like jump operation, roll operation for games) and precompiled primitives(existing libraries) as opposed to the traditional programming style where developers use a desktop/laptop based studio IDE tethered with a mobile simulator to test their apps.

This study brings together two ideas completely different and still in sync with each other. Studies have shown that touchDevelop makes it very easy for an app developer to simulate their code directly on a mobile phone emulator. It short circuits the entire procedure of first writing the code on a desktop/laptop based studio IDE and then testing it on an emulator. There are 20,700,000 scripts already running on the touchDevelop platform. These scripts are all open sourced and we have studied this data to relate the concept of social coding with the new platform.

The data analysis of the REST APIs showed us trends which were strikingly similar to the social coding sites like Github and had similar functioning as that of Version Control. In this study, we go through different trends and patterns to contrast and draw parallels between the leading version control platforms (Github, for this study, in particular) to TouchDevelop. Our study found that maximum updates were received by the scripts that were updated between 5 to 10 times. We found outliers with a script updated 25 times, found 200 comments. We did a similar study for reviews, we found scripts updated 15 times received maximum reviews, up to 8000 reviews. We also looked at the types of web abstract syntax tree available for different scripts. We found different nodes, and their comparison with most reviewed and most commented scripts. We found how different nodes varied between these scripts.

We also found the user platform distribution and the distribution of physical devices used mostly for the scripts made on TouchDevelop. We also found the top ten most popular users of TouchDevelop, their maximum reviews received, maximum comments received, showcase scripts if any.

We noticed TouchDevelop has 249,976 users, 700,000 base scripts (which can be called projects here), 198,765,900 updated scripts. The study on Social Coding for Github studies the project-project network, in a similar pattern as the script-script project we created for our study.

We also found the different user platform distribution for showcase-scripts, their physical devices and their cumulative positive responses.

Table of Contents

I Introduction

Mobile development has been on a rise for over two decades. There are Mobile development Platforms that integrate with cloud based APIs and include push notification for other users on that app. Firebase let's you concentrate on the front end development and handles all the hassles of setting up a key-value store. The Intel SDK also comes as a Chrome extension, knitting together different editors with a simulator. However, it is browser enforced and we need to traditionally hook up an emulator to get the right look and feel of the app. There are a couple other platforms like Icenium Mist and Parse which concentrate on push notifications and APIs that store data on the cloud. Sencha Architect comes the closest to touchdevelop where it lets you drag and drop widgets into an outline of iphone, kindle, Blackberry , Microsoft surface or custom sized screen. Then the framework handles layout, widget management and event juggling. It runs on a webkit browser and can be turned into an app by wrapping it with phoneGap or Cordova. All these applications have their own style of mobile application development but in theory, touchdevelop gives the fastest editor to directly simulate the app in your phone.

At the core of TouchDevelop is a typed, structured programming language that is built around the idea of using only a touchscreen as the input device to author code. The abstract syntax, typing rules, and basic semantics of the language are fairly similar to corresponding fragments of mainstream programming languages such as Java or C#. [5] As opposed to a traditional editor, TouchDevelop has a structure that is not entirely code-editing. The beauty of this Platform is that the semi-structured editing helps eliminate the possibility of syntax errors and users can concentrate on making the connection between events instead of employing time in compiling their code correctly. Over 207 million scripts have been written and downloaded via the touch develop platform. Fig 1.1 shows the TouchDevelop IDE as on safari desktop. It shows how the features are pre-compiled in to functionalities like the method set life(10) gives a player 10 lives without writing any additional code. In Fig 1.2, we see how we can select elements without having to type them. Every row is selected and then we get the available options on the bottom of the screen. We can select them, and expand the app with these options without having to type any number. This gives us a clear idea of how Microsoft planned to only have Touch as input for the IDE.

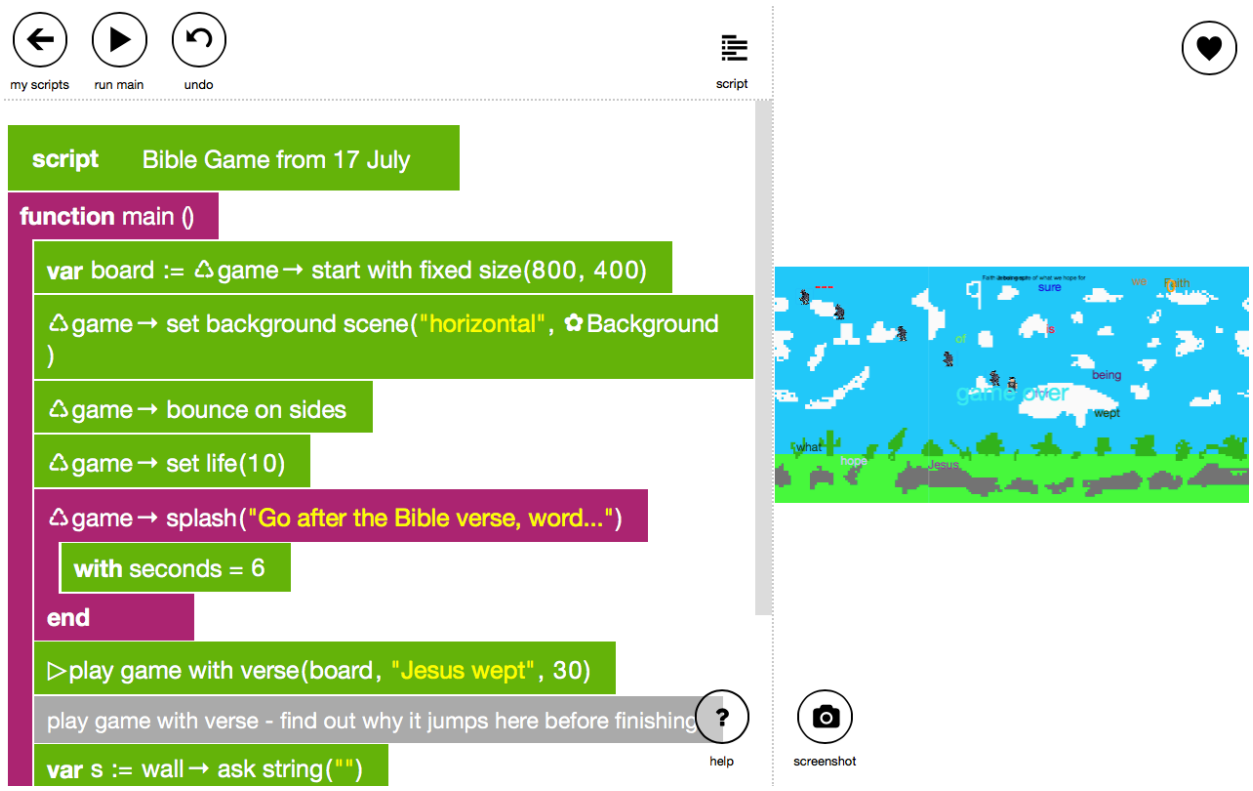


fig 1.1 TouchDevelop Editor as on safari Mac OS-10

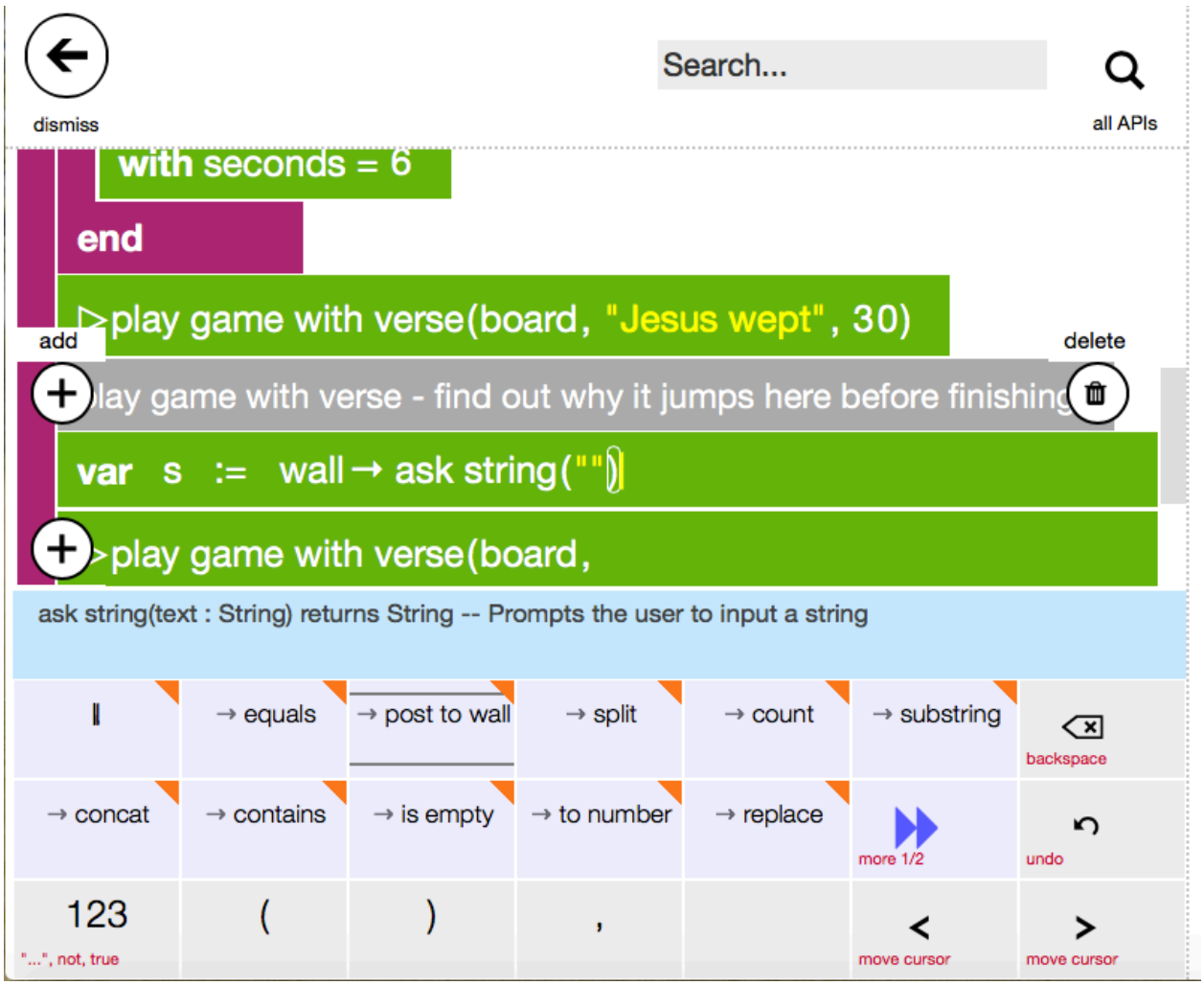


Fig 1.2 Shows the elements on TouchDevelop IDE on safari

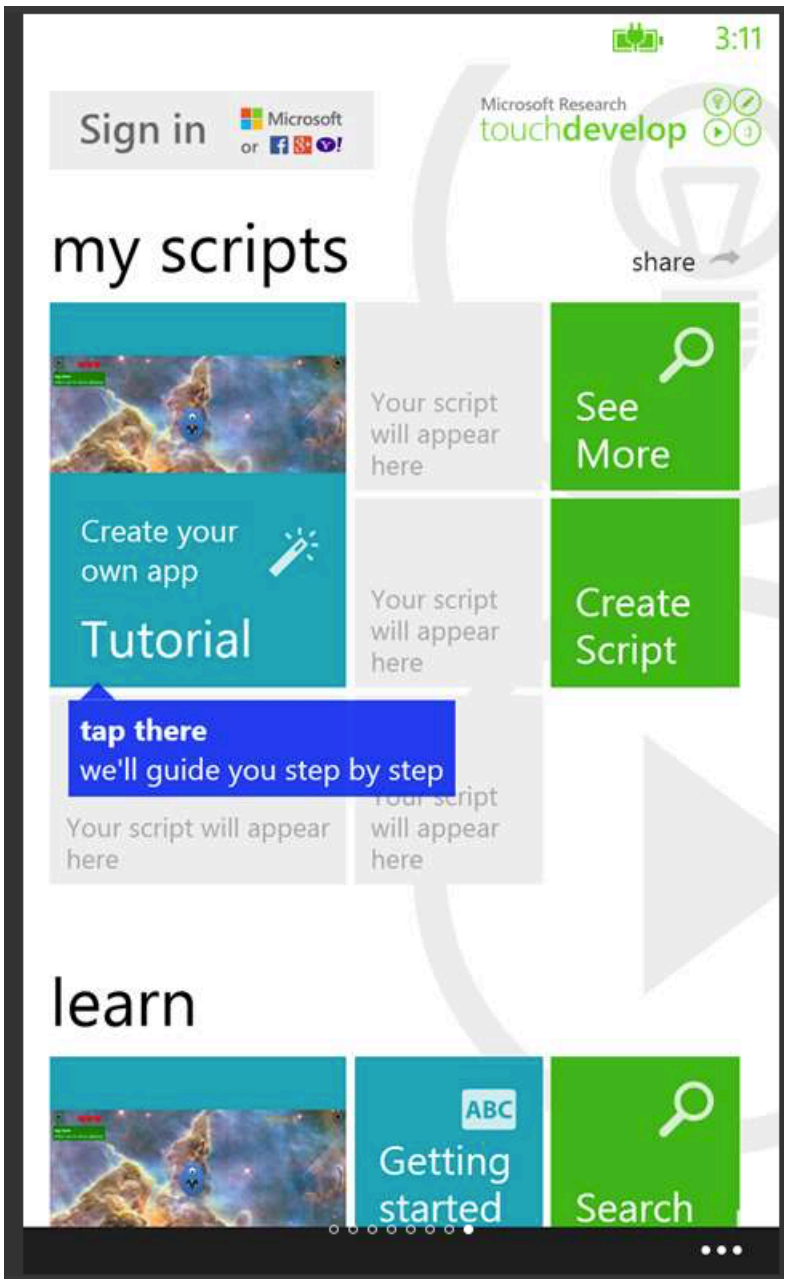


fig 1.2 TouchDevelop as on windows phone[5]

The editor was made in such a way that users get an experience of collaboration and are prepared for the professional world of working together. The app provides with the functionality of writing comments, publish screenshots, like a script if they like the work. Users can even download the scripts and/or add their piece of code to make the script more interesting. There are further ways to use phone sensors like GPS, cameras, accelerators that can further enhance the quality of the developed app. Users can also subscribe each other which is very similar to the GitHub functionality of following users. This gives a closer look at the core principal of the editor and the way the platform is supposed to give an all-encompassing experience of social coding.

In this study, we see different scripts and how they differ from each other. We notice how the social attention gives the user more motivation to make a better app. We also correlate different trends and compare that with the leading collaboration sites like Github to see if there is a similarity in the commits and/or difference in the way users collaborate on touchDevelop.
The study is divided into research questions which have been answered using the data that we have studied . The results are mapped using graphs and charts. We also compare Github trends with TouchDevelop data patterns, to see if we can get a similar trend with following scripts or users. The scripts that are most commented/ and or are most reviewed, have a pattern of being updated

more. We want to see if a similar culture ensues on the TouchDevelop app. Questions like these and more are targeted in this study to figure out the environment and the over all social coding norm.

Summarizing the major contributions made,

Mined and recognized common patterns in the touchdevelop scripts, drawing a parallel between the set standards of social coding available to us right now and what touchdevelop's data shows us. We have directly connected the social coding practices alive at the TouchDevelop IDE and how they are already practicing version control with out a use of a third party software or a plugin. This makes TouchDevelop unique in the sense of storing scripts while they are created and simultaneously syncing it with other scripts on the cloud. Currently, no other IDE available, for any language or type of development (desktop/mobile/cloud), has this capability. We show empirical proof that TouchDevelop can be treated as more than just a training tool/or for the purposes of small application development [6],[7]

II BACKGROUND

2.1 TouchDevelop

As per, Nguyen ,Sarkar and Csallner,[6] small application, with below average training in TouchDevelop, a student programmer who has prior Java and Android programming experience is still more productive in writing TouchDevelop apps on a phone than writing Android apps in a traditional PC-based fashion. Their work mostly revolved around finding the extent of the editor and the semantics of the scripts written on it. They compared the editor to the general practices of a coding language and independently studied the scripts as an object. They did not study the API data, instead employed a more experimental approach towards the application.

Tillman, Moskal [5] majorly introduced the concept and talked about how the touchdevelop editor was different from the other in-browser mobile application development editor. This paper serves as an introduction to the touchdevelop IDE and helps us understand the application a little better

Ball, Protzenko [7] wrote about how the BBC microbit initiation is supposed to work. It shows the TouchDevelop(TD) IDE in a stronger light where we can fathom the connection between microbit, which is a small computing device like raspberry pi and TD. This helps us understand that TD is capable of

creating small software codes which have a physical application as opposed to its application as only to construct games.

In 2016, Microsoft TouchDevelop collaborated with BBC Microbit encouraging students in the K-12 years to engage in the mobile app and exercise concepts like computational problem solving, games, robotic science and, of course, code. Several technology firms came together to endorse this concept where a small instructional computing device would be connected to the touchdevelop editor to make more effective apps and real-life applications. The software engineering hurdles to combat the design and architecture are explained in the 2016 publication stated in [7].

Code.org took up the charge to challenge the students to using a variety of online tools, and, also training teachers in the USA. In UK, CAS (Computing at School) created a curriculum for the two products to come together and that increased the touchdevelop scripts by 28% in a month alone [7]. The study was based off on the success of .Net Gadgeteer, which is a Microsoft product, apart from that the interest of raspberry pi and aurduno (open-source electronics prototyping platform), also generated the idea of combining the hardware and software technologies.

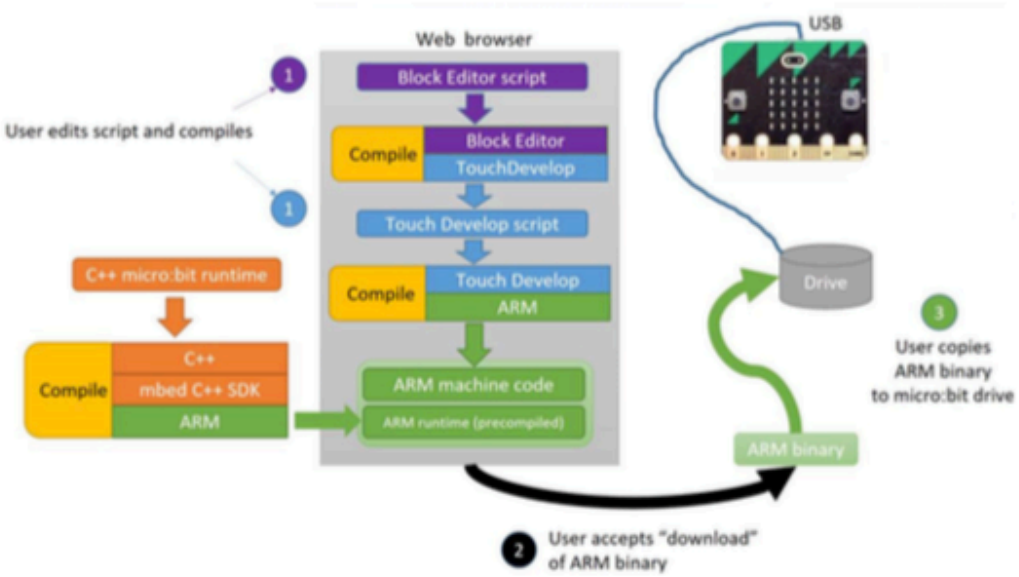


Fig 2.1 The compilation flow for TouchDevelop with MicroBit[7]

TouchDevelop has a root script which means that these are created from scratched and are not based on any other TouchDevelop scripts. These root scripts further become base to other updated scripts (which gives us the hint for version control, since the updates are made in different versions).

TouchDevelop has registered over 700,000 root scripts which were then updated more than twenty times by different users. Also, the scripts use the inbuilt library functions a lot, which will be shown in our study. This shows that users at TouchDevelop are enabled to use library features to enhance their scripts via tutorials, and/or looking at other scripts(which further confirms our hypothesis of the social coding culture) Without prior training, and/or technical background as mentioned in [6], TouchDevelop, has allowed us to visualize the

coding style of programmers who are not professional but are, budding in the same field. The users of TouchDevelop publish their scripts in cloud, they can also download the code, add to the script, favorite it, add comments and reviews. All this data is available on the cloud in the form of Rest API to the public. We explored these APIs to get a better understanding at first of the TouchDevelop users, the platforms and functionality. However, as we looked at the patterns, the data shows that users are meant to learn from better scripts (users subscribe to other users whose work they want to follow) which further reinforces our theory that Touchdevelop was designed to push the users towards the practice of social coding.

2.2 Social Coding

We have come a far way from the linear textual paradigm to a model based Structure, which means a software or a system that is modelled using the OOP techniques. The OOP language models are structurally rendered in a tree or graphical notion. This means that a certain domain model gives way to another, there are classes inherited by other classes and it is not necessarily a linear pattern. TouchDevelop has a similar framework, which is why it was easier to follow a pattern in the scripts. [8]

There are studies available that show that social coding enables a different experience of software development as the activities and interests of one developer are easily advertised to other developers [9]. According Softpedia Linux News, as of March, 2017,[16] Git is the most common source code management system and we have used Github numbers in our study for comparison for the same reason. Github, quite clearly, is a social network for coders where we get to exhibit and follow coding skills from different parts of the world.

Github has a high developer-developer network, project-project Network and a developer-project dependency involved.[3] This network has been subject to a lot of studies and we have used these studies to draw our own comparisons to a similar pattern that we have derived for the touchDevelop scripts.

We define scripts as the code snippets or applications written by users on the TouchDevelop IDE. These are available to us on The Rest API using the scripts Api. This shall be explained in detail in later sections. A script-script network would be a network of scripts connected via updation. A root script, as explained before is the script written from scratch, anything that uses that particular code(script) becomes an updated version of that script. It can be used to further enhance features on the same script or just use it as part of another script with a completely different functionality.

We have divided the touchdevelop networks into script-script network and user-user network. We use the pagerank algorithm to find the more popular users based on their subscriber history. This way it does not threaten the validity of our study as in [9]

2.3 PageRank Algorithm

The Page Rank algorithm was introduced in 1998 to allow us in calculating the more popular web pages on the internet.[15] This algorithm has been used by popular search engines like google to show the most relevant web pages based on how many other web pages refer to it. It is a straight forward mathematical approach to find the probability of anyone hitting a particular web page. The formula is given as under:

$$PR(v) = = \sum_{k=0}^{n} PR(k)/L(k)$$

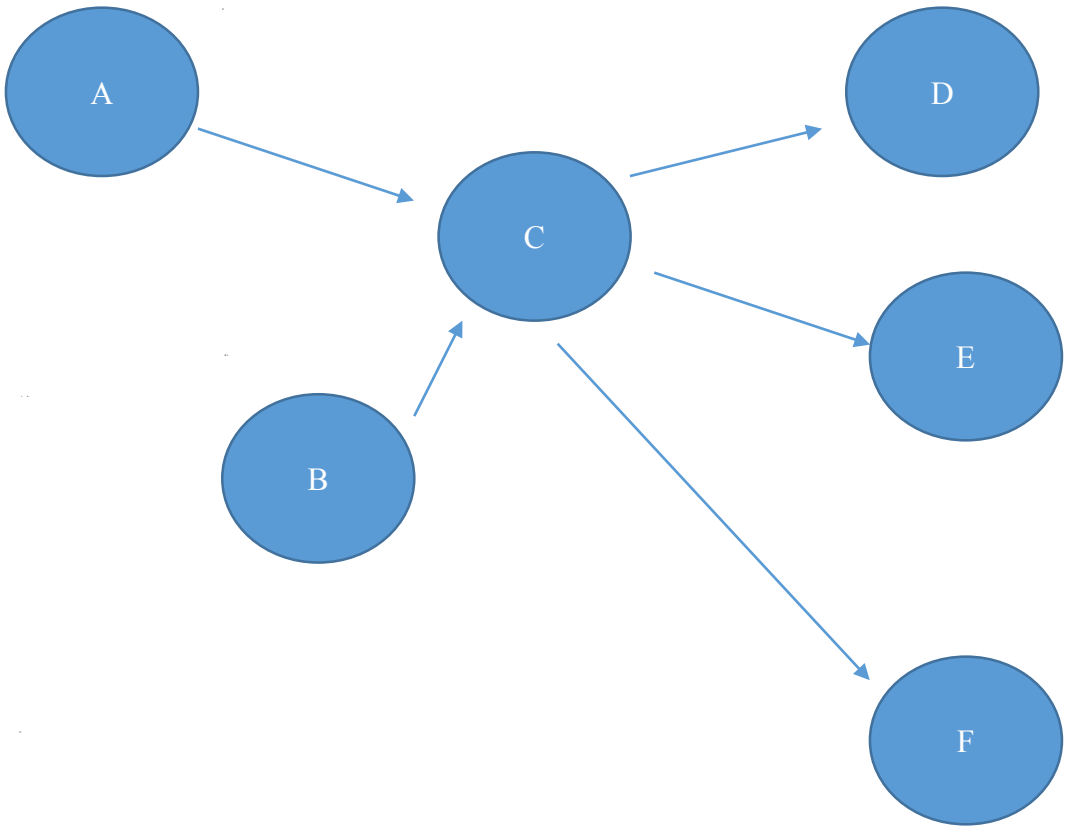
Where k represents the incoming links to v, whose page rank we are trying to decipher.

L(k) becomes the outbound link of the particular web page we look at.

The summation of page rank of all Incoming links divided by their own outbound links gives you the page rank of any web page.

This concept can be further explained with an example. Let's consider the model

below as part of the page rank network we want to target.



Now, Let's assume these are all the nodes that there are in the network, although There are many many nodes, for the purpose of explanation, we will assume,

these are all the webpages in the internet. We start with assuming initially that all pages have equal probability to be hit, hence the initial probability would be 1/total no of pages.

A	1/6=0.16
B	1/6=0.16
C	1/6=0.16
D	1/6=0.16
E	1/6=0.16
F	1/6=0.16

Table 2.1 Initial Value of the nodes

We see that C has two incoming links from A and B respectively, hence it would have the contribution of two web pages. However, the out bound links of A and B are only one each. This means their respective page ranks would remain the same (divided by 1)

Hence by the formula stated above the first iteration page rank of C becomes

PR(c)= 0.16/1 + 0.16/1 = 0.32

If we keep moving in a similar direction, we come up with the following table as their final page rank.

A	0.16
B	0.16
C	0.32

D	0.26
E	0.26
F	0.26

Table 2.2 Final page rank value of the nodes

A few points to note here would be:

- The summation of all the webpages' page rank totals out to be 1. It makes sense Because, it is after all the probability of hitting that particular page in all the Pages, and by that logic it should total to 1.

- Also if you notice that in spite of having two incoming links, the page rank of C is not much higher than that of D, E or F (which has only one incoming link each). This implies that the number of incoming links is not the only decider for a page rank to be higher. A web page can have a higher page rank if it is referenced by another high ranking web page.

- Finding out page rank of a particular web page is an iterative process. We need to keep doing the calculations till it stabilizes to a particular number, since there can be reverse links and page ranks can change with every iteration.

- In real life, there are dangling points, and loops which are handled by different algorithms, however we will assume for our study that we don't have any dangling points or endless loops.

Now, how do we use this algorithm to figure out our most popular user. This algorithm in spite of having a very dedicated usage in the internet has a model we can easily correlate with our graph mining. We created an input where the user-user network is the graph in question. We make the userids the nodes and the connection between them (subscription) as the edge. We use the same algorithm to check who subscribes a user and is further subscribed by another user. More details on exactly which algorithm is used will elaborated up on in later sections.

We have seen version control giants like Github, TFS and Gitlab come across similar patterns, which made for a very interesting observation. The pivot point of this study then changed to drawing similarities and whether that culture was only for trained professionals or it could also be extended to users who had little or no experience. The results are what built this study and have been compiled into a series of research questions and hypotheses. More about it is explained in further sections.

III Related Work

We are combining the research of two very different concept in this study, hence we will be tackling the related works of the two fields separately.

3.1 TouchDevelop (Earlier Empirical Studies)

A lot of work has already been done on the TouchDevelop mobile application Development editor. Most of this work is done by Microsoft research labs. However, there are other studies as mentioned in [5], [6]. As per, Nguyen ,Sarkar and Csallner,[6] small application, with below average training in TouchDevelop, a student programmer who has prior Java and Android programming experience is still more productive in writing TouchDevelop apps on a phone than writing Android apps in a traditional PC-based fashion. Their work mostly revolved around finding the extent of the editor and the semantics of the scripts written on it. They compared the editor to the general practices of a coding language and independently studied the scripts as an object. They did not study the API data, instead employed a more experimental approach towards the application.

Tillman, Xie, Li did a comprehensive study on the TouchDevelop scripts in 2013. The scripts were much lesser, only 17322 scripts were studied and 4275 users were recorded. [18] Our studies have taken into account larger numbers of

scripts and more user base has been involved. The questions that [18] answered were more to the nature of the structural features of TouchDevelop scripts like the Line of Code was studied, external library calls were also studied.

Another question that was answered was the kind of users that are on TouchDevelop. The user base was also studied with respect to ordinary users, experts or novices. Our study goes more in detail with the reviews, comments, length of the code, the way the scripts are written, how many times they are updated and their correlation with the reviews and comments.

The third question that was answered related to the code-reuse ratio of touch develop scripts, which answered questions like how many times a script was updated, or a certain script was reused. Our study also correlated this number of updates with the comments and reviews received.

Anderson, Li and Xie [21] conducted another empirical study on Touchdevelop, where they studied the game scripts look like when compared to non-games scripts, this comparison was done based on lines of code, frequency of use, and use of platform features. We did a study of more generic scripts and did not divide them between games and non-games. The popularity was measured between games and non-games. How they measure popularity is not clear. We did a similar popularity study but with concrete metrics of reviews and comments.

Athreya, Bahmani, and Diede studied the nature of scripts, how the scripts were changed and what the users asked in support forums. [19] Even though, this information is very important, it still hasn't covered everything. The updates of the scripts are independent and no correlation has been shown. Akhin ,Tillmann and Fahndrich studied the code similarity in TouchDevelop[20]. The root scripts and the derived scripts were studied independently. The Web Abstract Tree of earlier scripts was studied in this paper. The node types were individually studied. Characteristic Vectors were also studied in this paper of scripts, however, these studies were all independent and none of the studies actually correlated these characteristics of the scripts with the how popular these scripts were.

Tillman, Moskal [5] majorly introduced the concept and talked about how the touchdevelop editor was different from the other in-browser mobile application development editor. This paper serves as an introduction to the touchdevelop IDE and helps us understand the application a little better.

Ball, Protzenko [7] wrote about how the BBC microbit initiation is supposed to work. It shows the TouchDevelop(TD) IDE in a stronger light where we can fathom the connection between microbit, which is a small computing device like raspberry pi and TD. This helps us understand that TD is capable of creating small software codes which have a physical application as opposed to its application as only to construct games.

3.2 Social Coding

Microsoft came out with [1] where the Begel and DeLine suggested that social media is an important tool for collaboration and sharing information. [1] brings to light the impact of social media on new ways of forming software teams across the globe. *Web 2.0*, adds to the ubiquitous and searchable medium of the Web by making it easy for users to share information with the people in their social network. The programming community is growing more competitively than before. It also makes sense as all the brand marketing, publicity and sales are done on social media these days. It's not just about communication, but also, showing the technique that goes in to the product. The global perception to it makes the product more appealing. As new communities of software engineers organize into teams, they must communicate their aspirations and goals and come to consensus about their shared purpose. Online forums enable coworkers to discuss application ideas, future trends, potential markets, and their own personal requirements for working in a team.

Social coding has been gaining momentum and continues to as every application is being migrated from legacy to cloud. The whole idea is to make the procedure seamlessly less dependent on network, hardware and even software, like the Operating system. With this culture on the rise, it only makes more sense to collaborate in a social manner.

According to [1], [3], [4] Any size company, even one with a single developer, can

leverage pre-existing low-cost, low-overhead marketplaces to make their software available and visible to millions of potential customers. They can then employ social networking sites to virally market the soft- ware, first to friends and work colleagues of the company founders and employees, and from there to millions of potential customers who hear about the product from friends of friends of friends.

Feedback is another huge advantage of working in this manner. When scripts/code are posted online for scrutiny, a programmer opens his design pattern and analytical abilities to the online community. We have seen time and again, that subject to online criticism can be very critical. This helps the programmer to grow his skills and carve out their own corner of repositories where they show case their designs.

Apart from the right side of the V-model of software engineering, the developers learn a great deal about the left side too. Requirements and architecting the design is as important for product creation as implementing the design. The global collaboration makes this procedure much easier and much more effective with input pouring in from all sides of the world.

[3] and [4] talk about the devOps jobs after a coding experience where the continuous integration and testing is being collaborated on sites like Github. Jenkins, a popular CI/CD tool has github plugins to connect with so that they can implement CI/CD via github.[11] Examining the usage of microblogging sites like

twitter by software engineers, [10] tells us that they use twitter to communicate with their colleagues and source code, specification and design information were commonly shared over social media. Additionally, social media was used by more than 50% of respondents to communicate new ideas. [12]

[4] tells us that GitHub does not only help Communities of Practice create a shared understanding of their respective testing culture in peripheral and novice contributors. It also facilitates the diffusion of these practices among developers inside and outside of individual communities. Positive results of testing practices, such as adding features fast or badges with passing test results, demonstrate the *relative advantage* of those practices. As project owners strive to make it easy for new contributors to get started with their project and its test suite, they actively improve their project's *technical compatibility* with developers' existing practices. By lowering the barriers to entry — e.g., by providing existing tests, examples, and a working infrastructure for automated tests — project owners reduce the perceived *complexity* of their project's testing practices. Several integration and user interface features of GitHub support this, such as the built-in ticket system, external services like Travis CI, or the comfort with which code can be inspected using a Web browser.

With all this work, we can now conceive the imperative concept of social coding, and touchdevelop, being one of its kind in its style of coding, testing and sharing

apps, also caters to this methodology in a very subjective way. To study the patterns of touchdevelop and confirm it as one of the tools to bring forth in the mobile application development platform, the concept of social coding.

What makes this study particularly attractive is that open source continuous integration concept can be extended to touch develop easily. It refutes the claim that DevOps can only be done after years of experience, since users of touchdevelop with little or no experience, are also capable of mimicking the same methodology and it can be easily inculcated in programming styles for any developer.

Ⅳ Research Question(RQ), Expectations(E), Hypothesis(H)

4.1

RQ 1

What are the general features of a typical script developed on TouchDevelop. Are the scripts employing the inbuilt library functions, what mobile platform are we using to develop the apps.

Are scripts getting more comments and reviews, updated more often? Or vice versa, where developers pay more attention to the scripts if they know that they are being observed and followed? Or are scripts that are updated more often getting more reviews / comments? What we really want to know is if there is a correlation between the updates and reviews/comments.

H 1

On big social coding sites like Github, we have noticed that the trend is to show case the skills that a developer has. Apart from the correlation, we also want to see how many scripts used libraries, what physical devices were used mostly and what platforms were used mostly.

Since, touchdevelop has pre-set features, the touch develop scripts can only have more features involved in the script which means that web ast of an updated script will have more nodes than the root script. We have also compared progress stats between tutorials which are expected to rise with each tutorial.

However, since that information is not extensively available, we wouldn't be commenting on it.

E 1

The expectation is that the correlation of the number of updates, comments and reviews will be in direct proportion. We expect the graph to increase with updates, this is in proportion with our hypothesis where more updated scripts get more reviews and comments. However, we expect the trend to die down with increased number of updates, since the user base can lose interest in a particular script even after the script being updated many times.

4.2

RQ 2

If we find a script based on the maximum number of comments and reviews, does it give us the script with the maximum features and webast nodes?

H 2

As mentioned in 4.1, we have noticed that more comment and reviews can result into better scripts. The number of nodes in a script give us a rough idea of how many features have been implemented in the scripts. The number of comments and reviews are also part of the web abstract syntax tree, but we will be concentrating on the nodes that are apart from these.

E 2

The expectation is similar as that of 4.1, we feel that the study shows us that the most articulated script would have the most comments and reviews. It falls in line with our understanding of social coding and version control. The comments and reviews give us an idea of a more involved set of users and the script in question is more observed. This is what is implied by the norm of social coding. As discussed before, touch develop has shown us that social coding is a framework that can be easily extended to non-professional software developers as well. The user base of touch develop even though has little or no technical background, reacts similarly to the social coding norm. We will use this part of the study to prove precisely that.

4.3

RQ3

Are the most popular users the producers of the most popular scripts?

H3

Giants like Github have a user-user network, a project-project network, user-network dependency. [9] shows how that popular users have more articulated projects which are subject to higher observation, and in return present the best projects. Since, their entire reputation is built on their subject matter expertise, we expect better quality of product and better design pattern in terms of skill and

technique. We try to put the same understanding to touchdevelop scripts and find the popular users based on subscribers.

We calculate the top ten popular users of the entire touchdevelop user base using the page rank algorithm. We then compare the scripts of these top ten users and expect them to be in the results of 4.1

E3

We compare the user data with the script data and create a user-script dependency to check if we the popularity of a user can imply a popularity of the script. Conversely, we also expect the other way around to be true. We expect the popularity of a script to imply the popularity of it's user.

Here we run into some design conflicts. As any other popular social coding platform, one script is maintained by multiple users, these users could be amongst the popular users or not. Hence, we are not sure how that is going to turn out. However, we do expect the popular users to have one or more popular scripts.

4.4

RQ4

Does TouchDevelop have similar commit pattern as of Github.

H4

We have definite user pattern results from [3] and [9] regarding Github, the pattern of the user-user network, the project-user dependency and the project-project network. We would like to see if TouchDevelop can have a similar network, with users and projects. This is to see if TouchDevelop could be used like Github in future.

E4

Since we see similar pattern of commit, sharing and collaboration of scripts, we expect touch develop to give us similar results as of github project-user dependencies. We have noticed that a script that has been updated by multiple users can be treated as a project in Github. It might vary in terms of volume and commits, because we don't see a lot of updated scripts on touch develop but we expect to see similar patterns.

4.5

RQ 5

The special scripts of touchdevelop known as showcase-scripts are made to present the best work on the app to others? These scripts are picked by TouchDevelop on random to show in the link as shown in Fig 4.5.1 and Fig 4.5.2

Fig 4.5.1: Showcase scripts on TouchDevelop IDE

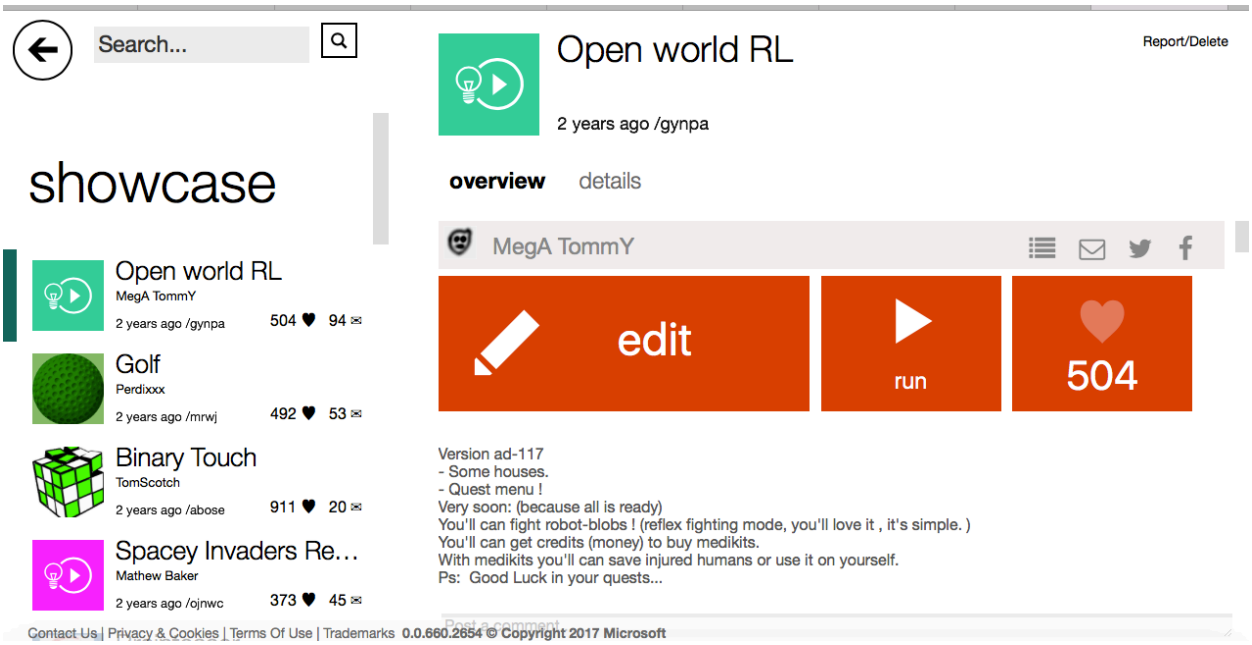


Fig 4.5.2: Showcase Tab on TouchDevelop IDE

H5

According to the REST Cloud API documentation by Microsoft, [13]The TouchDevelop API has information about the featured scripts like the user-platform, if any libraries were used, how many cumulative positive reviews were given to the script. Even these scripts are not many in numbers, they definitely gives us an insight on how we can use the touchdevelop platform to create the best scripts on the go. The expectation here is to get as much information on the perfect touchdevelop script as featured by the makers of the mobile application platform.

E5

These

featured scripts look exactly like the other scripts but they have all the vital information about the userplatform, what operating system was used or what was the physical device that was used to make it a possibility, were there any libraries involved, if there was a splashart involved, is the script tagged, etc. The normal scripts may or may not have this information recorded.

We will be making a list of all Operating systems used, divide the scripts into categories of libraries used, features and art used. This is done to figure out the showcase-scripts' built and make. These scripts are picked by the TouchDevelop team to showcase the best works on TouchDevelop. We also want to check what browser, since the TouchDevelop editor also has in-browser editor, is used most

often.

Physical device information is also captured in these scripts, and we expect windows phones to give the maximum platform numbers, since it is a Microsoft product.

In further sections, we will

present to you a more detailed idea of the solution and, how exactly are we studying the data with the information of all kinds of tools that we used to bring to you the data and visualize it.

V Overview and Design

5.1 Script-Script Network

We make a script-script network using the API data from the REST API given below:

www.touchdevelop.com/api/scripts

we use python scripts to gather data from this API, and return it in JSON format.

A sample script in the touchDevelop database looks like this:


```
{
    "kind": "script",
    "id": "rxiiiy",
    "time": 1509572644,
    "userid": "lmpwt",
    "username": "Alexis Shanks",
    "userhaspicture": false,
    "userplatform": [
      "unknown"
    ],
    "name": "Doggo Eats",
    "description": "This script has instructions (in the form of code comments)
for building the Chase and Gather (basic) game.\nID = TDR20b",
    "positivereviews": 0,
    "subscribers": 0,
    "comments": 0,
    "baseid": "aojti",
    "icon": "emptycircle",
    "iconbackground": "#9955bb",
    "cumulativepositivereviews": 0,
    "screenshots": 0,
    "rootid": "ubhkwvmk",
```

```
    "updateid": "rqvmnh",
    "updatetime": 1509572646,
    "ishidden": false,
    "islibrary": false,
    "screenshotthumburl": "",
    "screenshoturl": "",
    "mergeids": [

    ],
    "editor": "",
    "target": "",
    "meta": {

    },
    "iconArtId": "myrzglai",
    "raw": "",
    "scripthash": "e94d331dbb7be4ac2ace7fb4568e1d69",
    "updateroot": "rxiiiy",
    "unmoderated": false,
    "noexternallinks": false
  }
```


Fig 5.1.1 Sample script Json

The following tags are the ones that we will be using, and below is also the what information is gathered from each tag:

“id”: gives the script id, this is an important tag. We use script id to track root ids, in the updated scripts and it also tracks back the updated script to the original script.

“userid”: gives the user that worked on the script, please note this does not necessarily mean that he is the owner of the script, this only implies that the data saved as this script was written by the user to which this user id belongs. It could be an updated script for another script or a script that was further updated by

other users.

“Updatedid”: this id gives the next version of this script. In case a particular script was

never updated the “updatedid” would be the same as rootid.

It also has tags like ,”comments”, “reviews”, but to go through each Json to find all

comments and reviews seemed like a tedious task. Hence, we found the comments and

reviews through their own API. Samples of both are given as under:


```
{
    "kind": "comment",
    "id": "jvbzigflki",
    "time": 1509559540,
    "userid": "dggju",
    "username": "Mr. McMann",
    "userhaspicture": false,
    "userplatform": [
      "unknown"
    ],
    "publicationid": "lkeha",
    "publicationname": "my magic 8 ball",
    "publicationkind": "script",
    "text": "Kinda cool I guess but was kinda a little boring",
    "nestinglevel": 0,
    "positivereviews": 0,
    "subscribers": 0,
    "comments": 0,
    "assignedtoid": "",
    "resolved": ""
  },
```

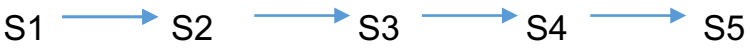
```
{
    "kind": "review",
    "id": "rnokvublss",
    "time": 1509571802,
    "userid": "tpiu",
    "username": "hi people",
    "userhaspicture": false,
    "userplatform": [
      "unknown"
    ],
    "publicationid": "jfnanl",
    "publicationname": "Bob the Blob: Space odyssey",
    "publicationkind": "script",
    "publicationuserid": "vzgt",
    "ispositive": true
  }
```


Fig 5.1.2 Sample Json result of comments and reviews respectively

we compare the publication id in both “comments” and “reviews”, to get an exact number of comments per scripts. We have then checked the updated ids to the root id to collect all the comment and review numbers for the most updated script.

This also gives us a better view of how the scripts are updated with increasing comments and reviews, answering our Research Question 1, where we are looking for comment to update ratio and review to update ratio. The formula used to find this ration is explained below:

Let’s assume there are five scripts, script 1 is the root script, then it is updated to script 2, then to script 3 and so on. The graph looks something like this,



Now, our first API, ~/api/scripts, tells us that a S1 has been updated 4 times, however, the number of times, we will find the rootid as “S1” will be 5, hence the no of updated scripts for any script comes from the following formula;

No. of times a script was updated = Dividing factor= No of rootIds found -1

Now, in the comments API, which we get using ~/api/comments, we might have different comments for different scripts, we will find the “publicationId” with the comments, we count comments for all scripts that have been updated,

hence, the total number of comments would be

total comments for scripts = summation of comments in individual scripts

the ratio is now found by diving the total comments with the diving factor.

This ratio, as is expected should always be greater than 1.

We will plot a graph between the comment to update ratio with root scripts for all root scripts.

This way we get a general idea of the how a script is updated with increasing number of comments.

We will also show a couple of scripts, and the difference in the number of comments with each update.

We do a similar thing with reviews and plot graphs for the same.

This takes care of our research question 1, now coming on to research question 2, we are calculating a script with the highest comments, for this research question, we are not considering the previous scripts, we are considering the comments that pertain to only that script. We take the top highly commented and reviewed script and check the updated ids and web ast of the script.

The web-ast will be give us the features implemented in the script. More nodes will give us more feature. Something to note here is that both comments and reviews are considered as nodes, but we are careful to remove these nodes from our calculation.

Conversely we are also going to check the script with the highest number of Nodes and see if that script has enough comments and reviews to put up with the standards of social coding. The idea is generating a relationship between a developer's interest in the script against the attention that script receives. We are hoping, it will be in direct proportion

5.2 User-User Network

Touchdevelop has over 100,000 users constantly using the IDE to make mobile development data on the go [5] , especially after they collaborated with microbit and expanded their market, there are more and more users everyday. We created a user-user graph with the help of subscriber information in the user API. The user information is available on the www.touchdevelop.com/api/users link. A sample user Json looks like as under:


```
{
   "kind": "user",
   "id": "pfumi",
   "name": "Tiffany Minter",
   "haspicture": false,
   "time": 1509587357,
   "about": "",
   "receivedpositivereviews": 0,
   "subscribers": 6,
   "isadult": true,
   "avatar": ""
 }
```


Fig 5.2.1 sample Json of the user

We can further find the subscriber user ids by looking at the list of Jsons returned by the subscriber API. The subscriber API looks like this;

www.touchdevelop.com/api/”userid”/subscriber

a subscriber user id would look exactly like a user id as shown in 5.2.1, we can keep calling this API to find more subscribers in an iterative manner till we hit the

leaf node of the subscriber tree.

We create a tree of subscribers and iterate through the tree backwards to find the most popular user id, i.e. with the most subscribers or the most popular subscribers, as dictated by the page rank algorithm for the same.
When we get the top ten users in the sample data that we use, we compare the scripts and if their quality of work is reflected in the results of RQ 1 and RQ 2.

Since the user base of touchdevelop is very huge, we have tried to make smaller samples, to get a better idea of the pattern. The users are then compared and we present the number of scripts against the users, their number of updated scripts against the users and the collaboration of these users in different scripts, which clones the idea of projects on Github. Our study is trying to parallel the Github patterns as closely as we can.

5.3 Github Comparisons

According to [9], The developer-developer network was created to figure out the shortest length path and diameter of certain projects on Github. According to the paper, about 1,161,522 had a common developer between them. We try to find a similar pattern between scripts, however, our concern is also comparing the commit history of the developers. Github has developer guide data, [9] gives us details on the most sought after developers on Github. We can easily track the

commit history of these developers to get a clearer scenario on the website and compare that with what we find in our touchdevelop API data.

The develop-developer network was made in a similar that we have done with our user-user network. The only difference is that the authors of [9] assumed that if two users were involved in the same project, there was an edge between them. In our case, we don't have any assumption, on the contrary, we have enough validity while creating our edges because we take the subscriber information from the subscribers of the users. Which means that they clearly follow or subscribe a user and that gives a stronger support of popularity than just being connected via a project.

The page rank algorithm is used in this paper, similar to our approach in finding out the most popular user. Studies have only shown the pagerank of top 5 users for their samples, however, we do more than that, we compare their scripts. This kind of result is what we are expecting out of our study.

Another interesting study that came out in [9] was the edges between projects, which implies the common developers between the projects and how they collaborate which is essentially, the central idea of social coding. We intend to find similar results with our user-user network and hope to align it with the github results, if not in volume, at least scalability.

5.4 Showcase-Scripts

The featured scripts in touchdevelop are given in the showcase-script API. The API data is given by the following link:

www.touchdevelop.com/api/showcase-script , these scripts are fully equipped for data mining, and all tags have information for us to understand the platform better. We are looking at the user platform most importantly; we want to check the percentage division between different operating systems, different physical devices and different models of the devices. We noticed in the data that there are repeating entries of operating systems, whether it is a phone or a tablet, what physical device was used. We also look at the data of the number of positive reviews, whether or not a library was used to construct this app, or if there are any external links, whether it is touch or mouse (in case it was tested on web).

These scripts are all single scripts, which means they are all different apps altogether, however, that does not mean that these scripts are not updated at all. They have information for “updatedId” and “baseId” which implies that are a part of other apps but there is no repeating entry.

This information should give us a better idea on what makes the scripts optimized on touchdevelop, how do we go ahead and make use of the editor to its fullest capacity. We will be presenting this data in the form of pie charts, bar

VI Results and Observations

6.1 Script-Script Network

6.1.1 Comments v/s Updates

As discussed in section 5.1, we have created a script-script network which gives us the updates of a particular root script. There are 700,000 root scripts. After running the API data we found, 1,987,659 updated scripts. We found direct correlation between comments and updates, also, we did another correlation between reviews and updates.

Are scripts getting more comments and reviews, updated more often? Or vice versa, where developers pay more attention to the scripts if they know that they are being observed and followed? Or are scripts that are updated more often getting more reviews / comments?

We had expected a direct linear correlation between comments and updates. We had expected a similar pattern for reviews and updates.

We got the following graph for Comments to Update ratio, when plotting a line graph.(fig 6.1.1)

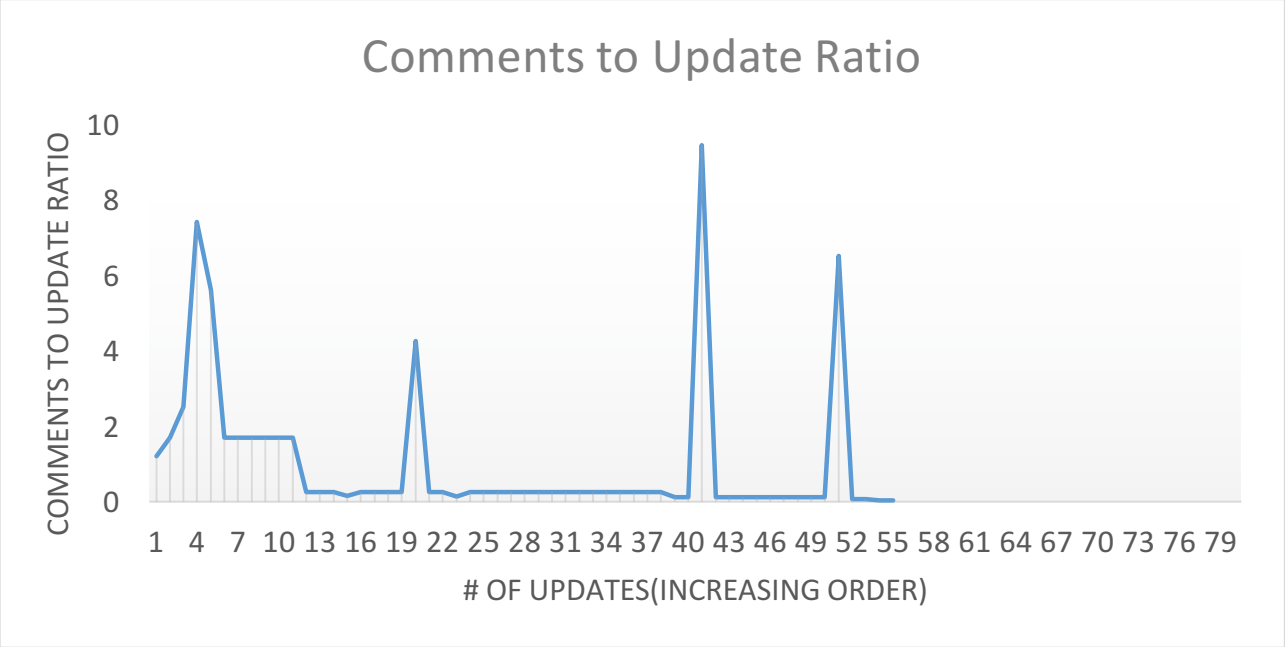


fig 6.1.1 Comments to Update Ratio

We notice that most of the scripts have a ratio greater than 1. This means that the number of comments is always greater than the number of updates. However, there are times, when this ratio drops below 1. This could mean one of two things; either the updates were greater then the compiled comments, or, the comments were lower than the updates. To see this comparison more closely we ran the numbers again, to plot a graph along with the # of updates (fig 6.1.2)

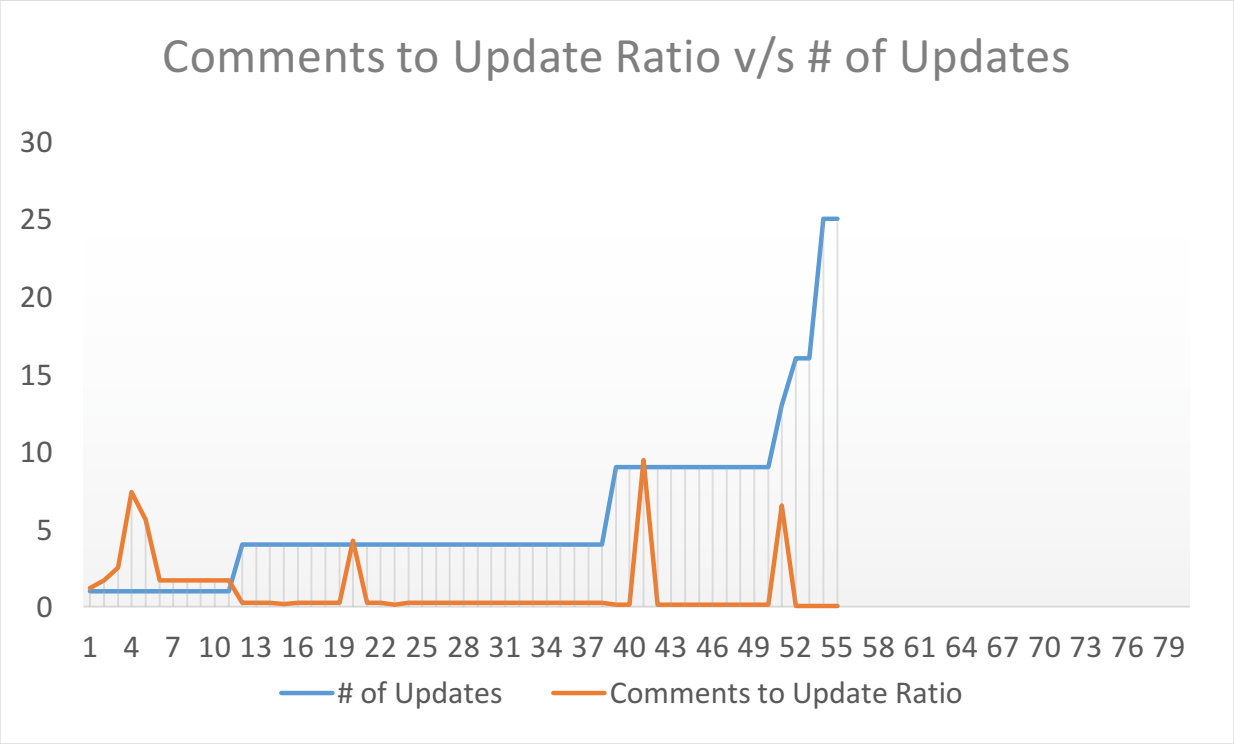


Fig 6.1.2 Comments to Update ratio v/s # of Updates

Fig 6.1.2 gives us a clearer picture of why the ration may be low or high with respect to the number of updates. Now we notice, that as move towards higher updates, the ration is still dropping for a lot of scripts, which helps us conclude that in spite of updated scripts, there were not a lot of comments. This doesn't entirely give us the exact measurement of the attention that those scripts got. However, for the ease of this study we will assume that the comments directly represent the kind of attention a script received. Mostly, however, the comments are in direct proportion with the updates and this takes us to the expected result of the social coding practice of keeping the script updated, if it received comments.

To further analyze the data that we received, we have also tried to show a distribution, which is not perfect, of the comments and updates. For this, we took our scripts and using their update number. Which means if xyz was the fifth update of abc, we calculated how the comments increased or decreased for that script. We ran this logic through all the scripts that we have and came up with the following box plot. (fig 6.1.3)

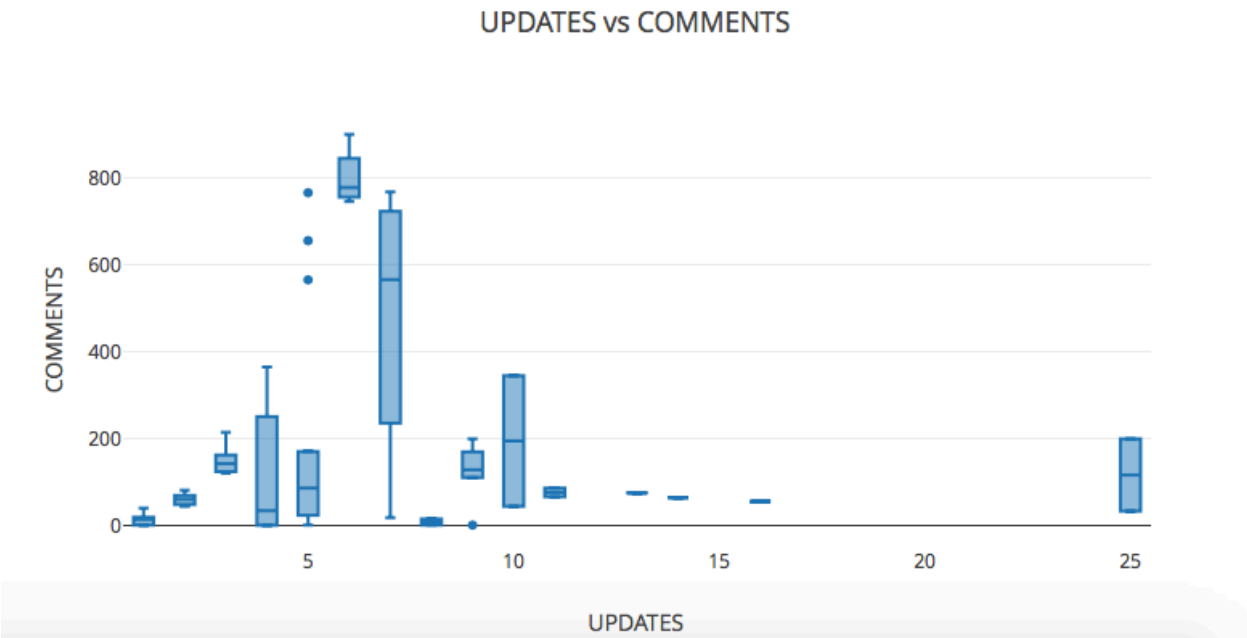


Fig 6.1.3 Box plot of comments vs Updates

Fig 6.1.3 gives us the information about how a script progresses in terms of comments versus their updates. The plot is created in a way that every fifth update is captured and their comment data is plotted in the box plot. This gives

us a better idea of each update for an evolving script. We can see that scripts that were not updated more than five times have also gotten good amount of comments. This shows that it is not mandatory to keep the script updated for it to grab attention. However, if we look at the 6th and the 7th update box, we notice that the concentration of comments increases in volume, with the maximum going off to beyond 1000 comments.

At the same time, I would like to point out that, if we notice the 25th updates are not doing so well with respect to comments. A reason for that could be that there in general, less scripts that have been updated 25 times. Also, it is possible that people lose interest in older scripts no matter how many times to update it. We need to remember that maximum of these scripts are games, and small phone apps.

6.1.2 Reviews v/s Updates

We did similar study on reviews, and plotted the reviews against the updates. As mentioned in section 5.1, we calculated the reviews to update ratio by calculating the total number of comments divided by the total times a script was updated. We got the following plot (fig 6.1.4)

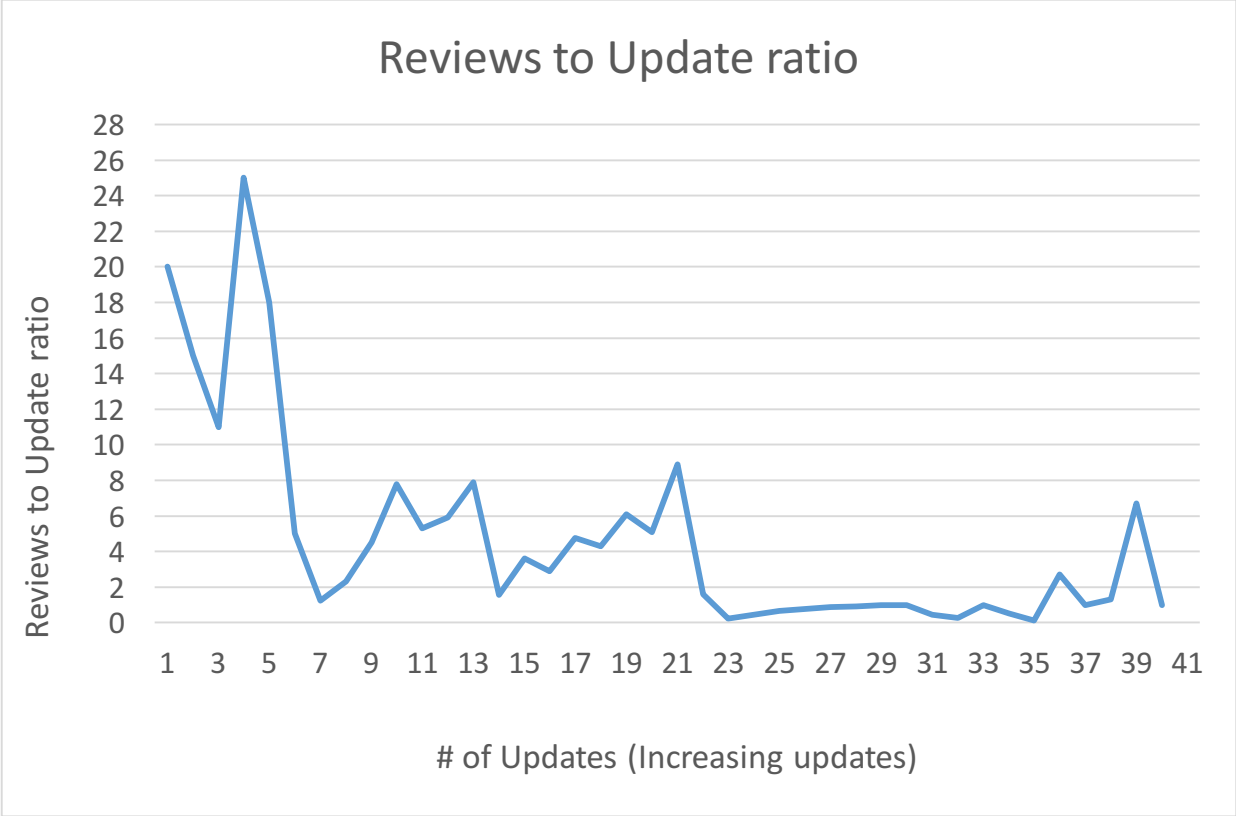


Fig 6.1.4 Review to Update Ratio

Fig 6.1.4 shows us that the reviews to updates ration increases pretty linearly, until it fluctuates in the middle, still remaining more than 1 for most cases, and then eventually going below 1 for a few scripts. Again, to get a better picture, we plotted the same graph with updates. (fig 6.1.5)

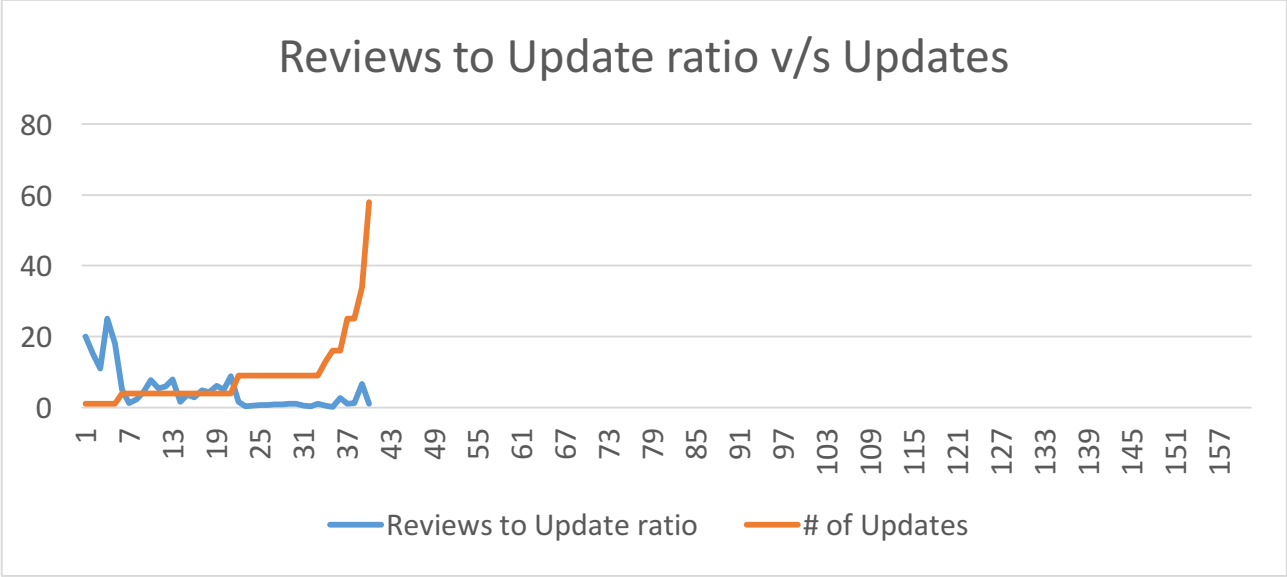


Fig 6.1.5 Reviews to Update ration v/s # of Updates

6.1.5 shows us a completely different picture, for all 7000 data points, this ratio is not directly correlated to the number of updates for lots of scripts. We can clearly see that with constant updates, the reviews are not necessarily increasing and in fact as we move right, we notice that these updates are increasing to a larger extent but the reviews remain low.

Something fascinating that came out in this review graph was that not all the largely updated scripts had comments on it. Since in 6.1.1 we saw that we got updates till 25 updates, however, we ran the reviews API, we saw that we also captured data for a script that was updated more than 30 times.

This says that if we have to look at popularity, Reviews are a stronger parameter than comments, since it covers more base.

To find the distribution of reviews, we plotted box plot (fig. 6.1.6) for reviews to see how every updates script fared with respect to reviews.

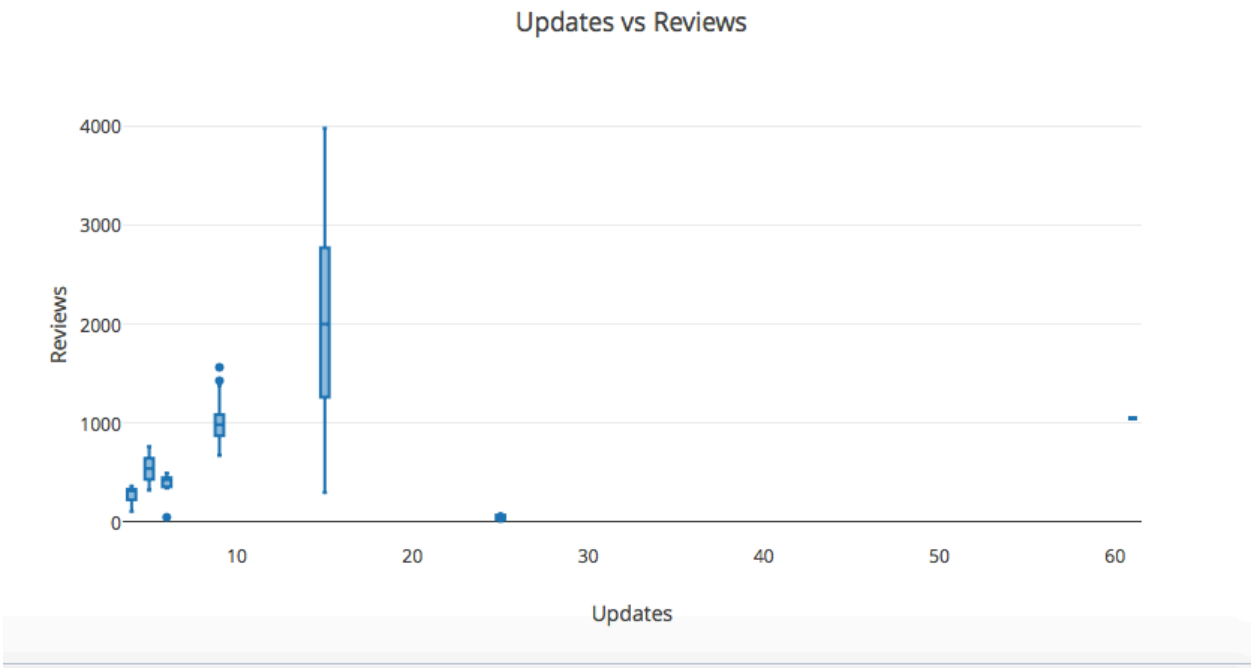


fig 6.1.6 Box plot of Reviews v/s Updates

Fig 6.1.6 shows us that the maximum reviews are given to scripts that were less than the 15th update. Maximum reviews were given to the scripts which were either the 15th update or the 16th update with maximum reviews of 4000. There are some outliers with respect to both reviews and updates. We notice that the 25th update have less than 500 reviews, which is very low in comparison to the other reviews. Also we have the 60th update, which has low reviews but this

could also mean that there are not very many scripts which were updated that many times.

6.1.3 Miscellaneous script information

We also compared the length of the scripts, between the biggest scripts, the most commented scripts and the most reviewed scripts. We plotted the graphs between different prominent type of node types. The plot is shown in fig. 6.1.7

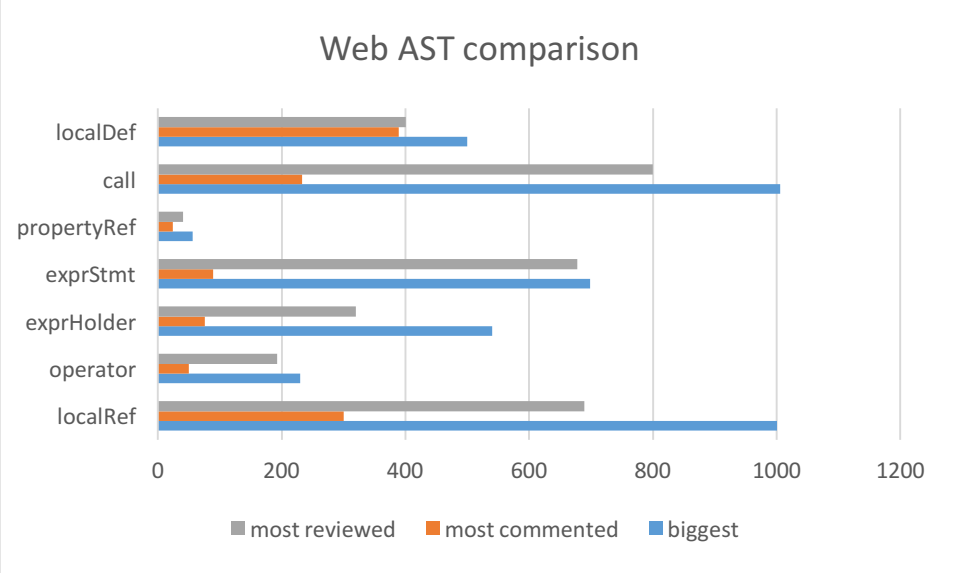


Fig 6.1.7 comparison of different scripts

Fig 6.1.7 shows us the number of nodes in different types of scripts, and also, a more specific distribution with respect to the types of nodes. We can see that the maximum number for a local definition nodetype goes a little above 1000, however, the most commented script has a little less local definition nodes than

the one with maximum nodes.

Similarly, for other types of nodes.

This answers our Research Question 2, which talks about the largest script with respect to the most popular script (most commented/most reviewed).

Since, the numbers are close to each other, we realize that the updates may have been in close proximity with the comments and reviews. However, the webast only shows us the complexity of a script and that could be the result of less updates as well.

This only shows that the script that got maximum attention on touchdevelop was articulated well in terms of complexity, with large library functions, and node types.

To get a deeper understanding of how the scripts are written we have done further analysis of some nodes. We have studied them against updates and review. Fig 6.1.8 shows the trend with Local Ref.

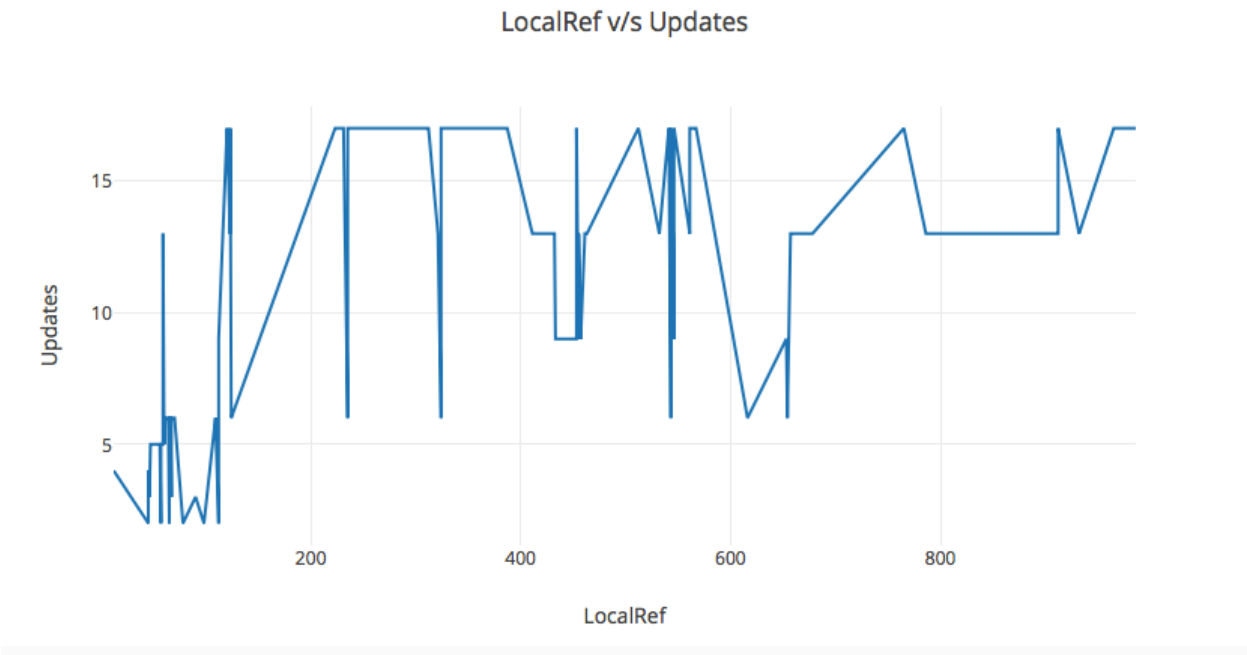


Fig 6.1.8 Local Ref v/s Updates

The figure 6.1.8 shows that local references with increasing updates we have more local references. This shows higher code writing intelligence. We also plotted against reviews to see how that affected the number of local references. Fig 6.1.9 and Fig 6.1.10 show the local ref v/s reviews and local refs v/s comments respectively.

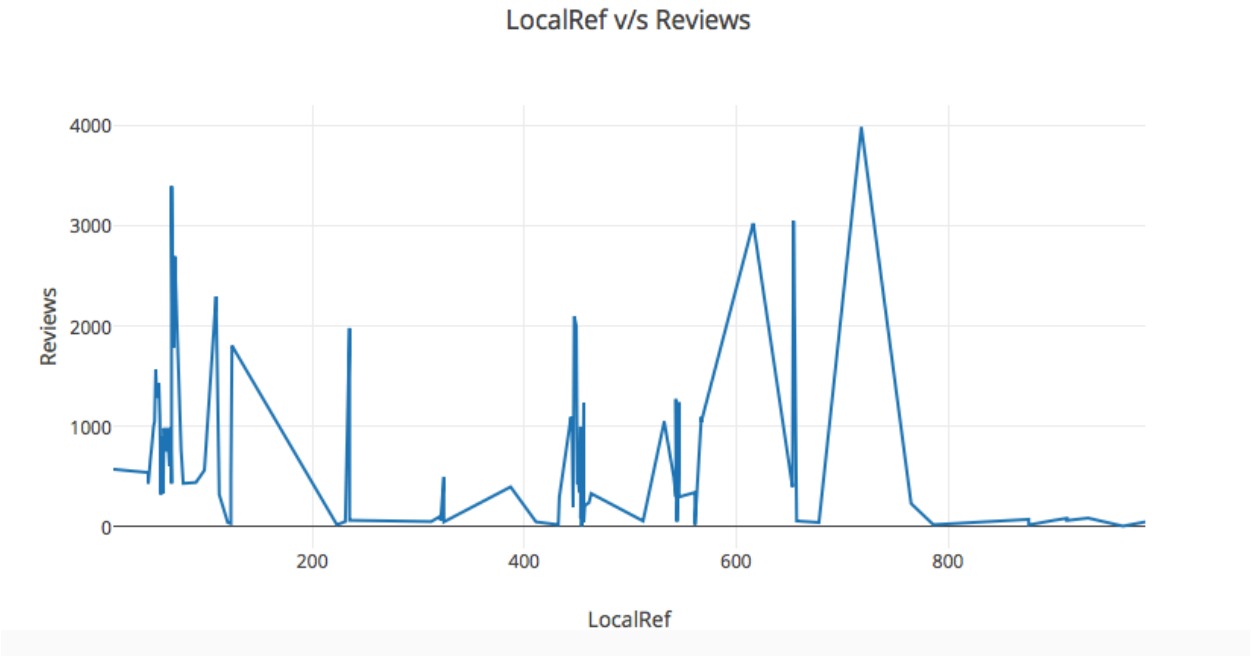


Fig 6.1.9 Local ref v/s Reviews

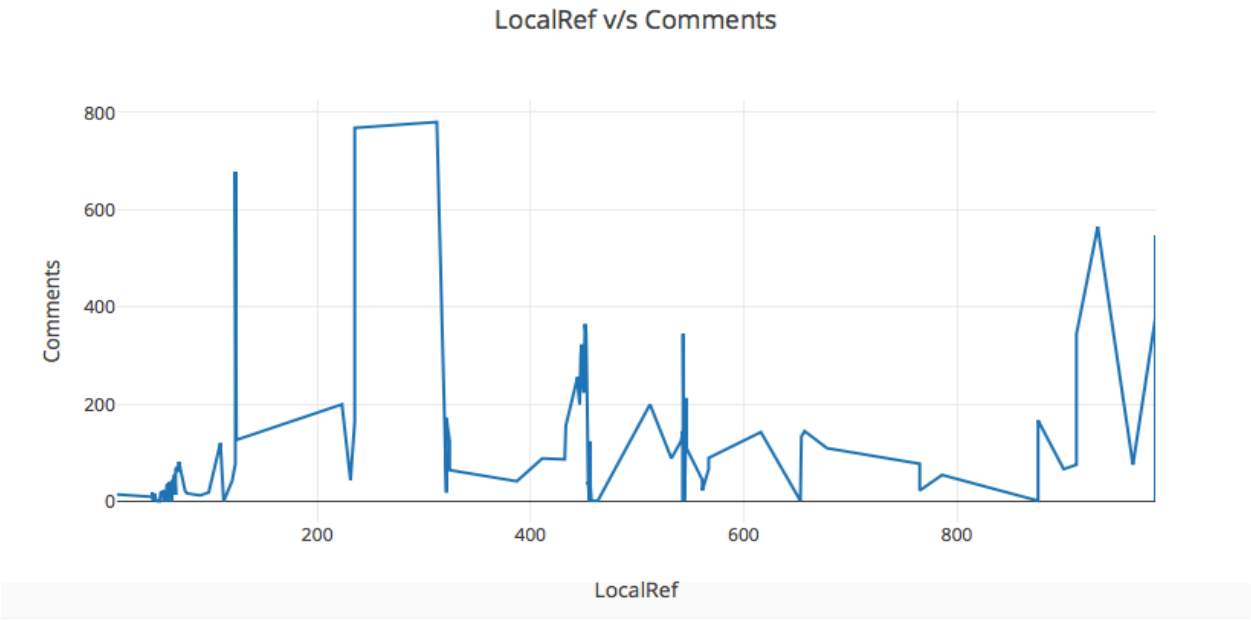


Fig 6.1.10 Local ref v/s Comments

We plotted Expression statements which include the logical and arithmetic statements. These statements contribute to majority of script given how the app expects touch as input, it is easier to make expression statements. Fig 6.1.21, Fig 6.2.22 and Fig 6.2.23 show the trend with respect to updates, comments and reviews respectively.

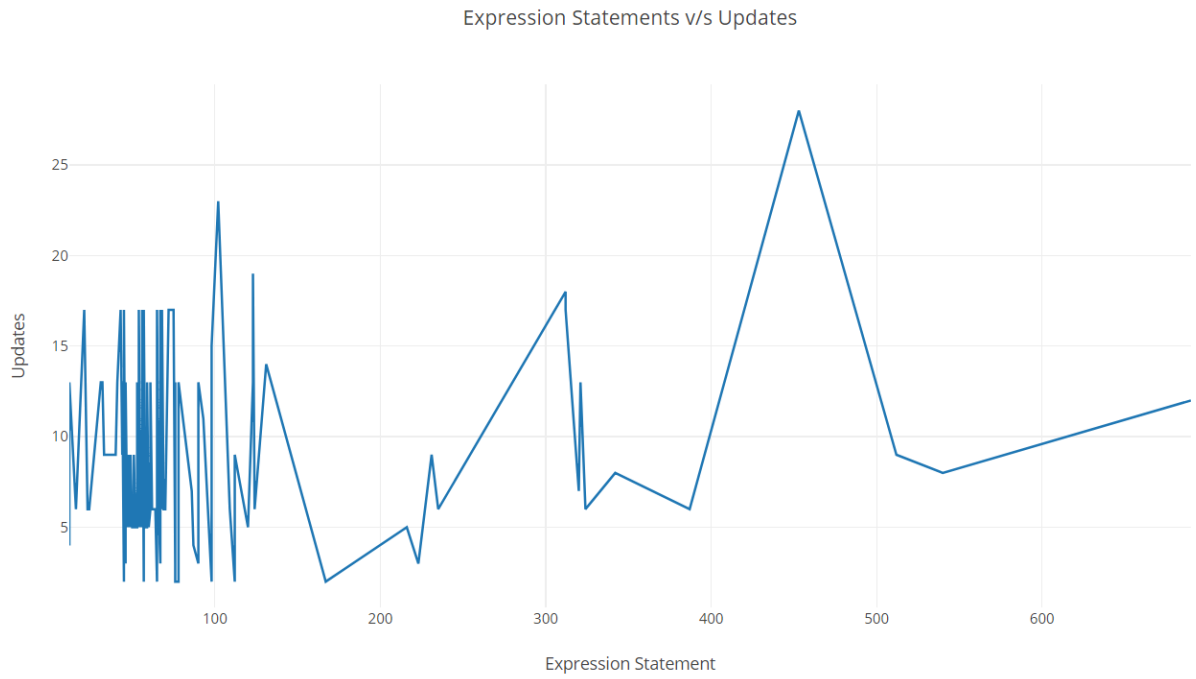


Fig 6.1.21 Expression Statement v/s Updates

The Fig shows that larger updates don't have more expression statements. However, we see an outlier at 450 expression statements is a script with about 25 updates.

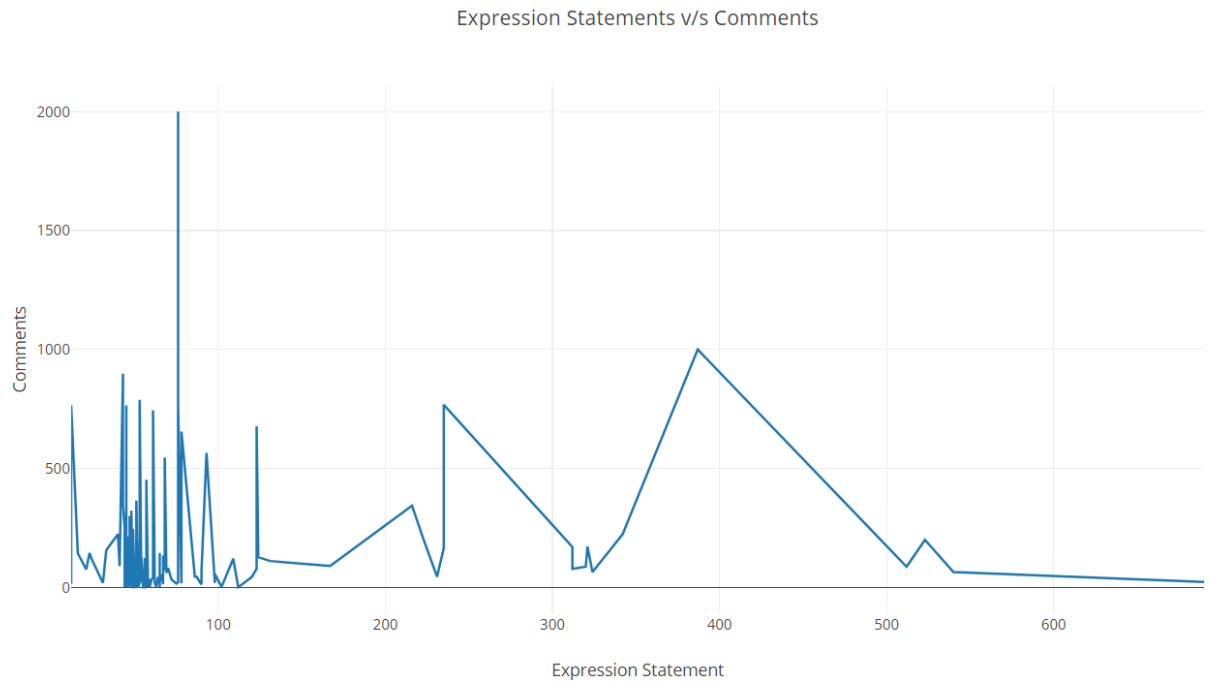


Fig 6.1.22 Expression Statements v/s Comments

This figure shows higher expression statements don’t account for higher comments. The popular scripts were fairly low in number of expression statements.

We also plotted Expression Statements against reviews.

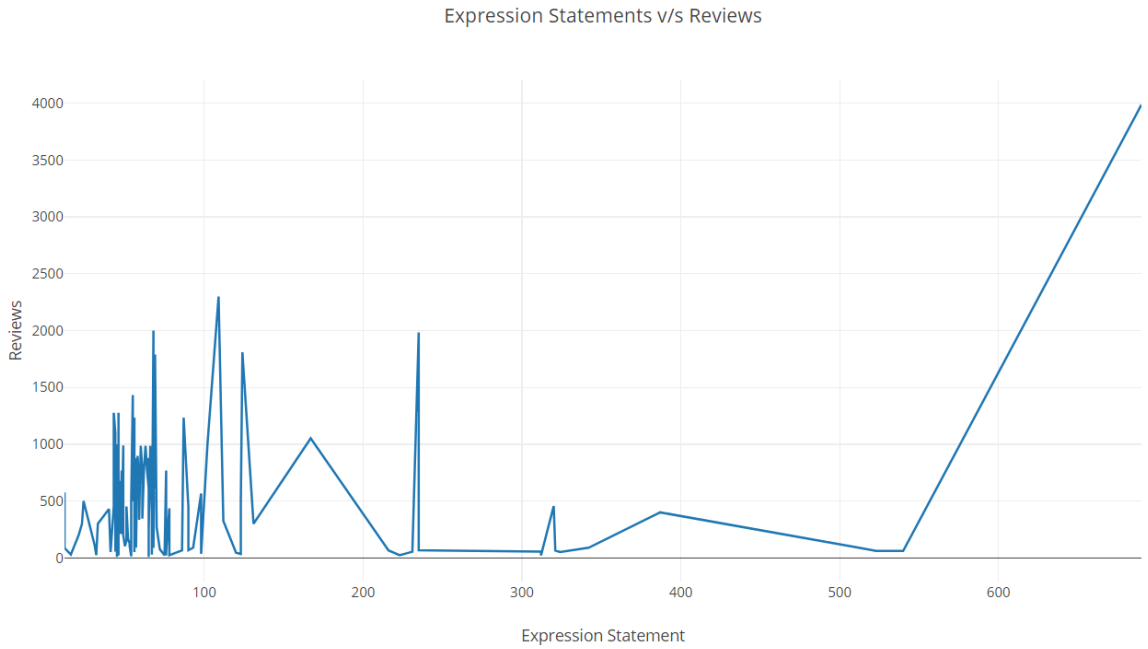


Fig 6.1.23 Expression statements v/s Reviews

Fig 6.1.23 shows us increasing reviews have more expression statements and that longer scripts, i.e. more complex games get more reviews.

We plotted calls v/s updates, reviews and comments. Fig 6.1.24, 6.1.25 and 6.1.26 show these plots.

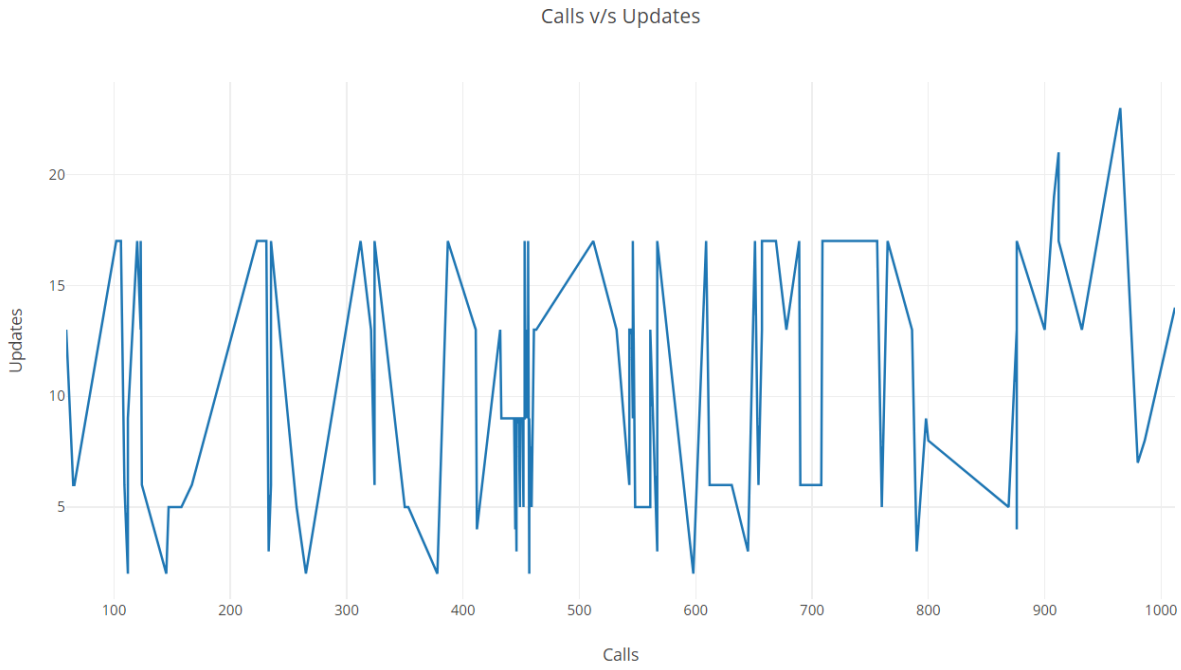


Fig 6.1.24 calls v/s updates

Fig 6.1.24 clearly shows us that the number of calls increased with the number of updates, there are times when lower updates still showed more calls, but over all it is a very steady increase between calls and updates.

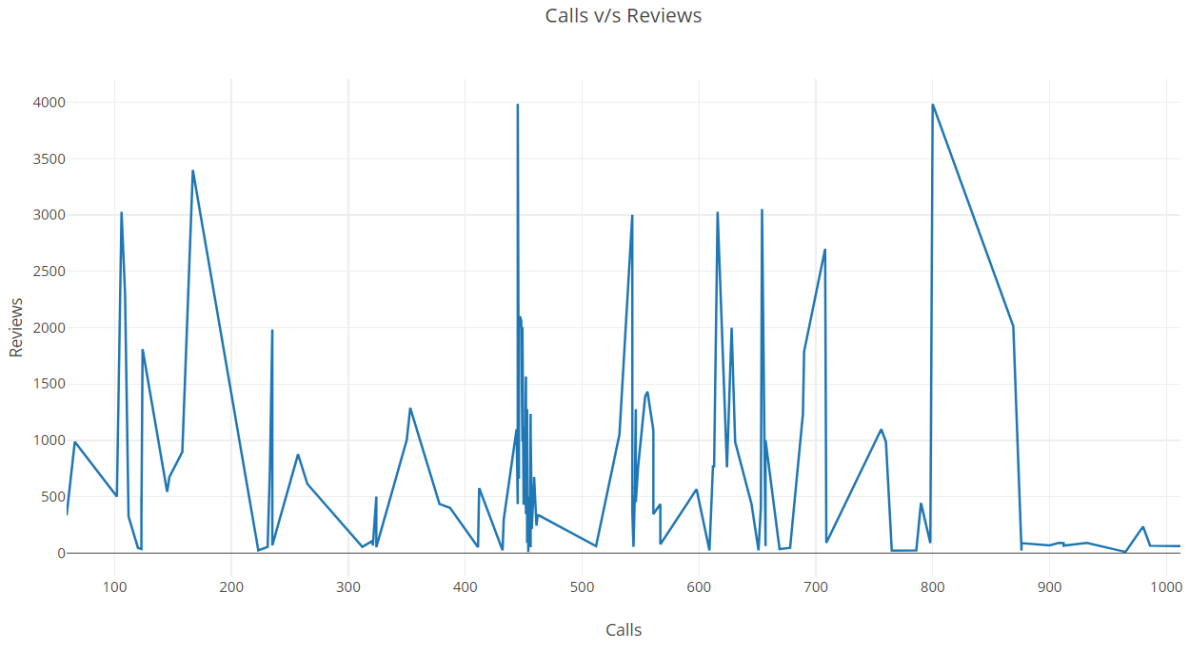


Fig 6.1.25 calls v/s reviews

The calls increased with reviews ans especially between 800 to 900 calls, the reviews went to about 3500-4000, which means greater calls gave greater likes(reviews). We also see an outlier where a script with 450 calls got about 4000 reviews.

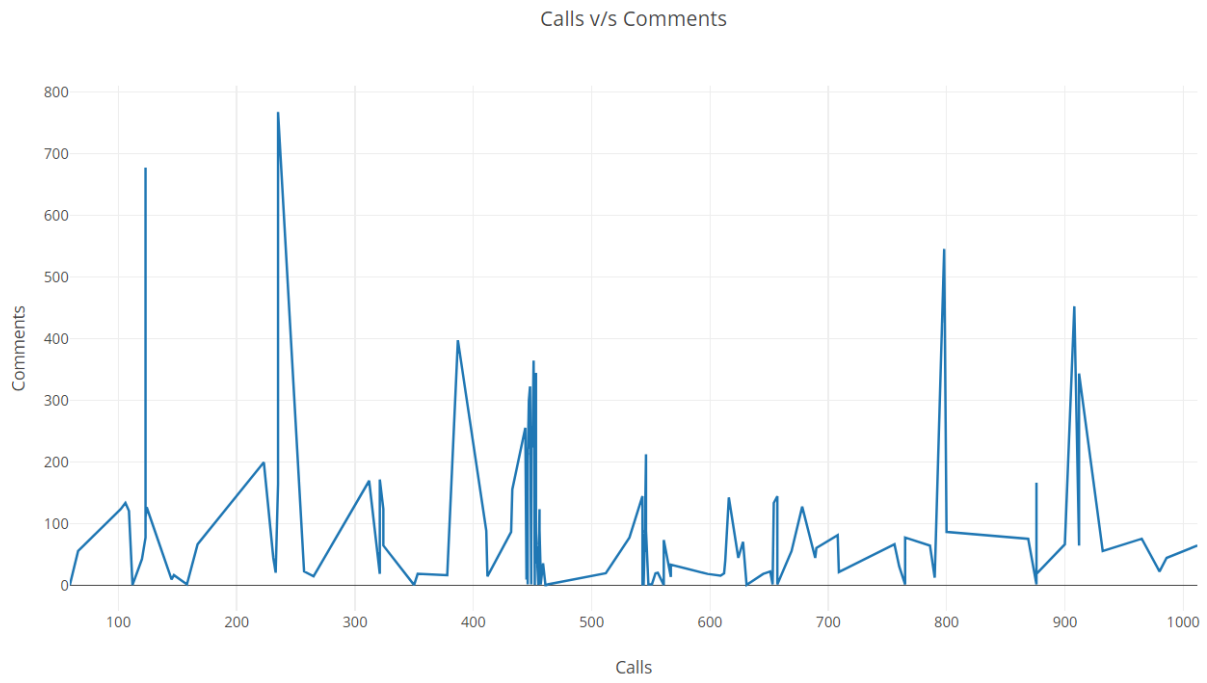


Fig 6.1.26 calls v/s comments

This pattern shows that code with more calls didn't get too much attention. That is not a definitive conclusion as the complexity of the games/apps have not been considered. However, general trend shows that the scripts got more comments with around 300 calls.

Apart from this, we also ran the scripts to know some other general properties of the scripts like the platform that they were built on. Fig 6.1.28 tells us the platform distribution of the scripts in general.

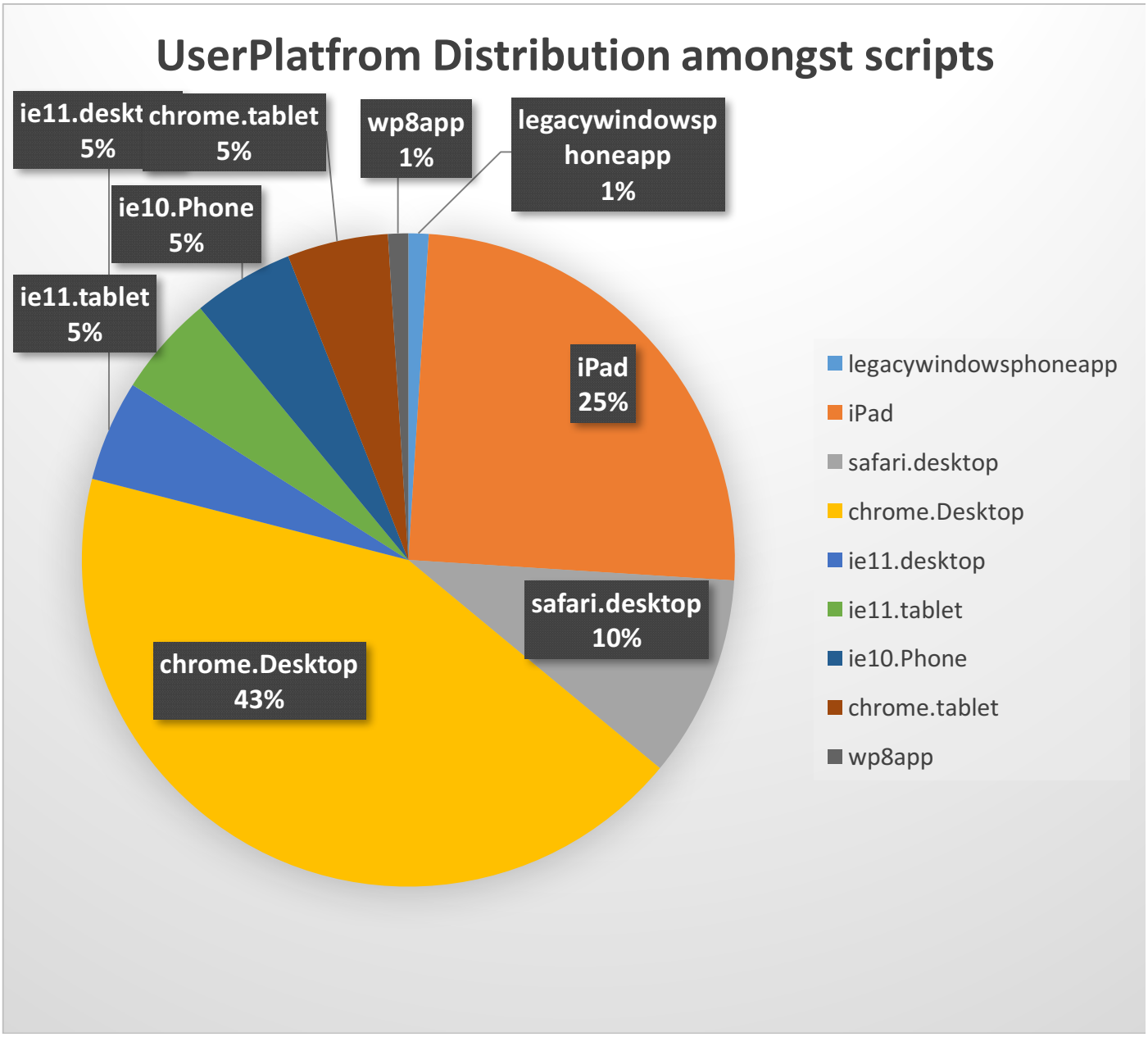


Fig 6.1.28: UserPlatform Distribution amongst scripts

Fig 6.1.8 shows us that maximum apps on touchdevelop were built on Chrome desktop, this might have nothing to do with the TouchDevelop IDE per se but can be because of the geneal popularity of the browser.

It should be noted that in the figure above, we can see only windows phones

(different OS versions) because the touchdevelop app only released for the windows phone. However, the idea of using touch as an input comes out successfully here because we can see that a quarter chunk of the developed apps are on the ipad using a desktop nevertheless.

The legacywindowsphone has a minor 1% of the apps developed on it. Something to point out here would be that a lot of apps didn't record the userplatform. Hence this number cannot be the final count, however, whatever data was recorded was enough to get this analysis. We did a similar search for physical devices used, and we got the following plot. (Fig 6.1.29)

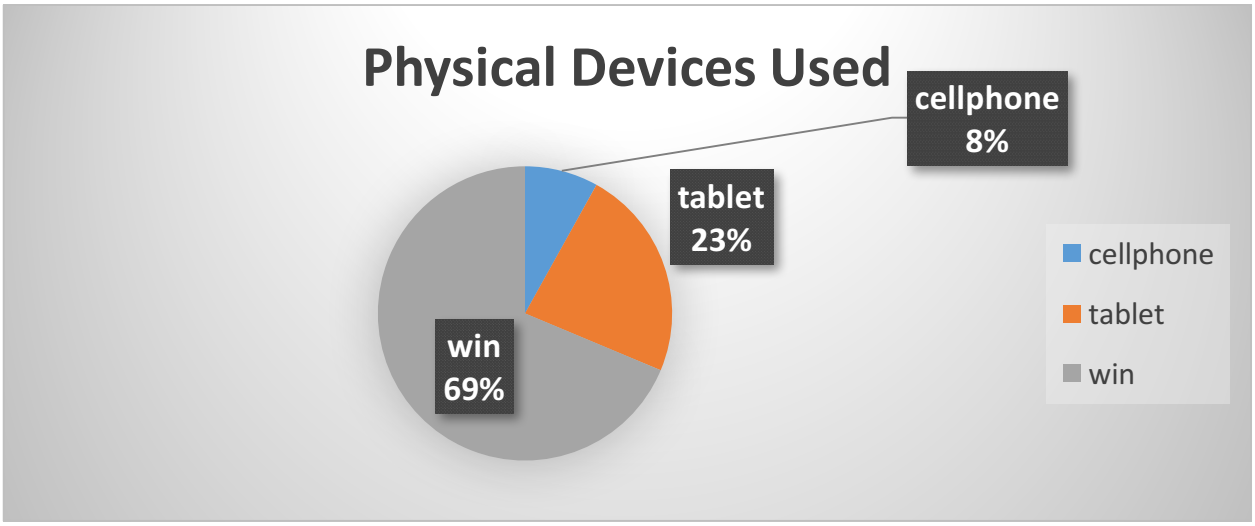


Fig 6.1.29: Physical devices used in the TouchDevelop Apps

Fig 6.1.29 tells us that majorly the apps were made on windows platform, a fair assumption that can be made here is that this was the in-browser TouchDevelop editor that was used on windows OS. However, since we saw earlier, Fig 6.1.8,

that a lot of scripts were created on IPads, we can safely say that the motive of having touch as the only input for creation of mobile apps, has been successfully implanted within the users of touchdevelop.

Again, these number are not the strict numbers, since we couldn't capture the physical device information for every script. However, we captured this information for 1,678,956 out of 1,994,659 individual scripts.

6.2 User-User Network

6.2.1 Most Popular User

We applied page rank algorithm to find the most popular user on the TouchDevelop app. As mentioned in 5.2, this algorithm was used as a graph mining technique to reach the most subscribed user in our case. We made this graph by putting users on the nodes, and if they subscribed to each other, there would be an edge between them. This made a directed graph, which was branched, we used page rank algorithm to iterate over this graph iteratively till we got a final value of all subscribers and calculated the most influential or subscribed user.

We found the total number of users on TouchDevelop were 249,976. These users also had some users who hadn't subscribed to any other user and/or were not subscribed by any other user.

According to the page rank algorithm, that we implemented, we found the most popular user to be with the id, ‘dgrys’. The user has written 46 scripts, with more than 500 nodes in the webast of the scripts combined. However, his scripts are not in the top most commented or most reviewed scripts, which gives us more ideas about the subscriber culture online.

Github has registered over 24 million users in 2017, with over 67 million repositories. [20] TouchDevelop users are very less in comparison but at the

same time, the subscriber-subscriber network is very similar to that of Github. There has been a similar study [3] where Github users were page ranked on the basis of the projects that they were involved in. Our study is definitely more direct. The reason is we are directly basing influence on subscribing the users. We found the most popular users on the touchdevelop with the maximum reviews received, maximum comment received and number of showcase-scripts. (Table 6.2.1) We found the top user didn’t receive the maximum reviews and comments, and didn’t have any showcase-scripts. This tells us that TouchDevelop is a very social peer depended app where the apps might not be popular across the IDE but still has a large network of subscribers. However, we also noticed that the top ten users included some users with around 6 show case scripts and 1838 reviews. This told us that it worked both ways, much like any

other social collaboration site where the popularity of your work and social influence both made you popular.

Username	Subscribers Rank	Maximum Reviews received	Maximum Comments received	Showcase-scripts(if any)
dyrgs	1	234	51	0
fdeo	2	162	134	4
cqeba	3	541	76	5
umey	4	543	21	0
Zpol	5	1838	45	6
Exil	6	1033	98	3
wvyva	7	355	102	1
gbpy	8	1174	213	2
nwjc	9	1254	34	1
nitj	10	164	12	1

Table 6.2.1: User popularity according to page rank in Touch Develop

We understood that if the network could be studied exactly like Github, the network of TouchDevelop was the same and that we could draw a parallel between Github and TouchDevelop. We also made a few more observations about the version control Giant and TouchDevelop.

As of 2017, Github has 24 million users, over 15 million repositories.
We noticed TouchDevelop has 249,976 users, 7000 base scripts (which can be called projects here), 1,987,659 updated scripts. The study on Social Coding for Github studies the project-project network, in a similar pattern as the script-script project we created for our study.
This aforementioned study also used page Rank Algorithm to get the popularity ranking of the users based on what users worked in the same project.

We have enough evidence to believe that the social coding pattern of collaboration is very similar to Github.

6.3

6.3.1 Showcase scripts

These scripts are featured by TouchDevelop. They are the not many, 271 to be exact. We also mined these scripts to see what these scripts are made of.
We got the following graphs for User platform, Cumulative positive responses and Physical devices. The plots are shown below.

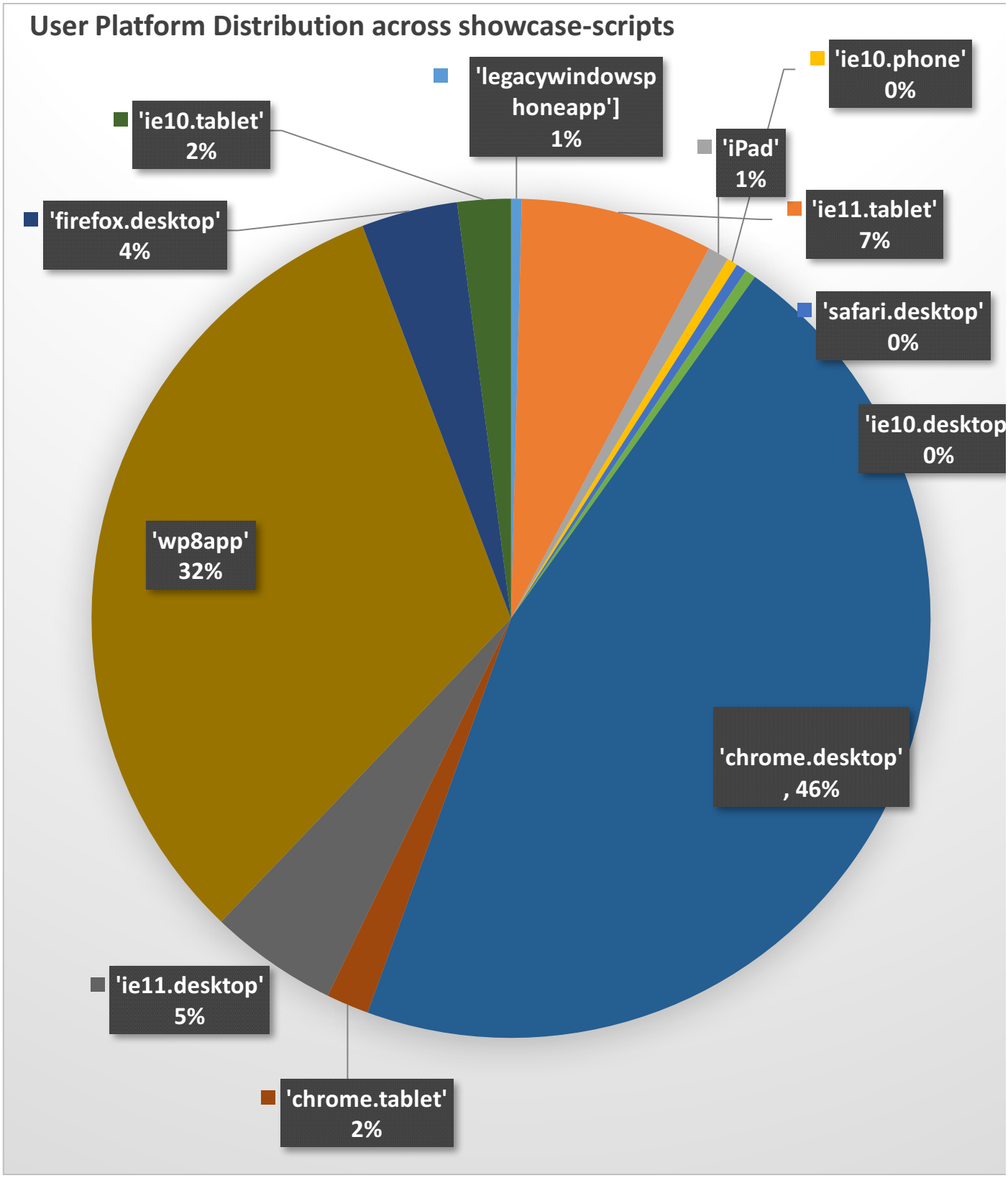


Fig: 6.3.1 User platform information for Showcase Scripts

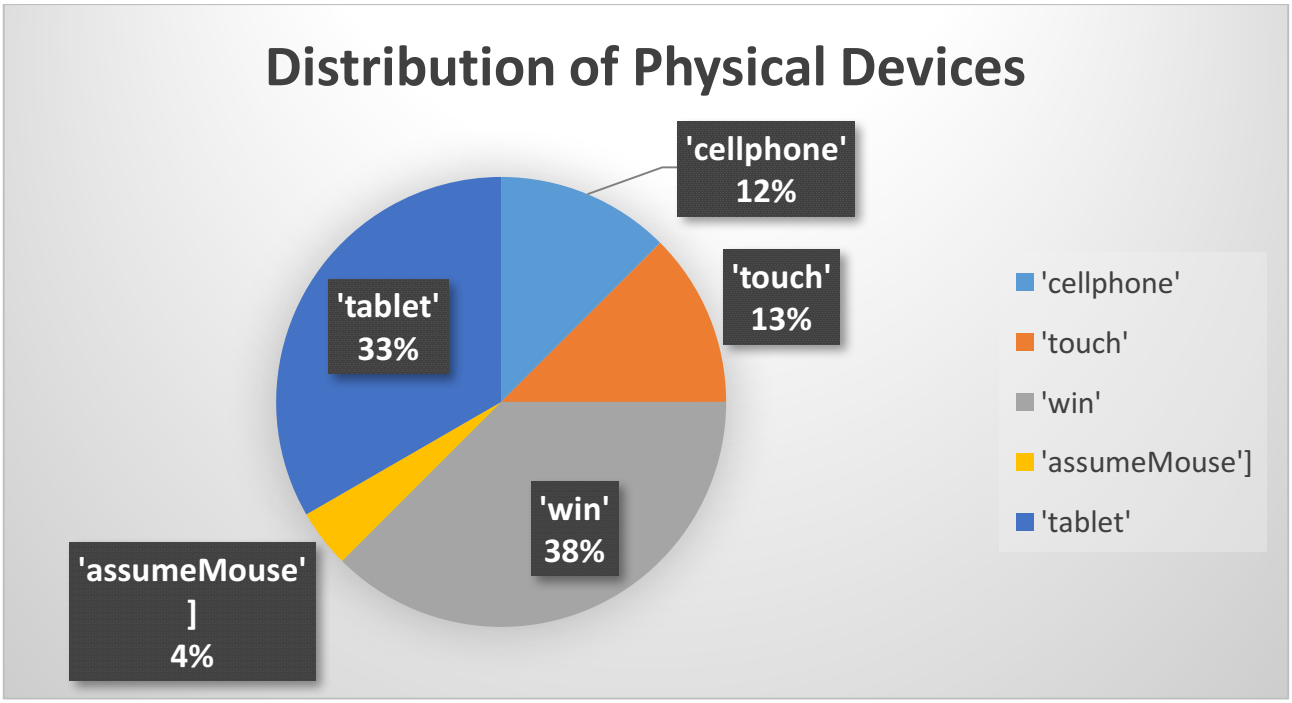


Fig: 6.3.2 Physical Devices on Showcase-Scripts

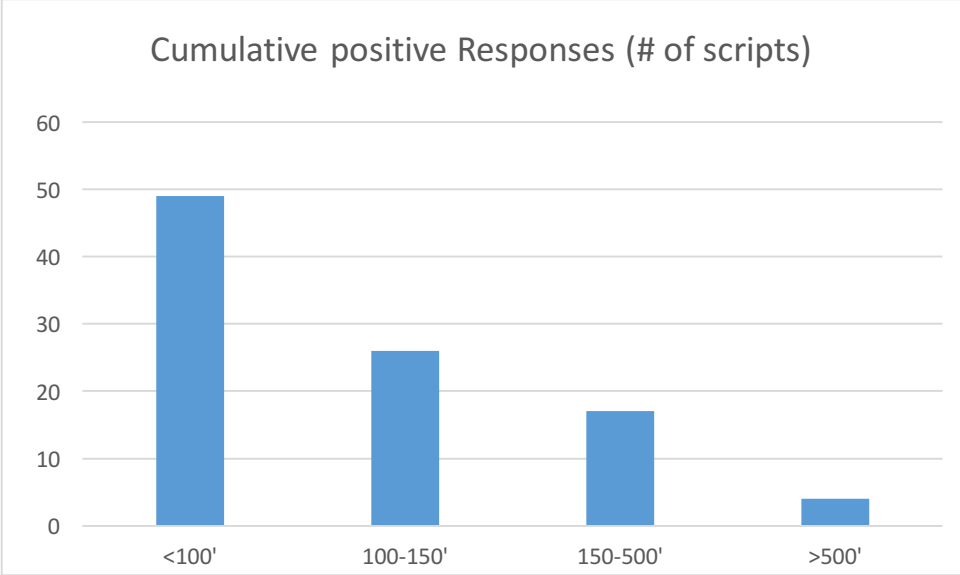


Fig 6.3.3 Cumulative positive responses on showcase-scripts

graphs and other visual aids so they are easy to understand.

VII Conclusions

There is a definite correlation between the reviews/comments and the updates found for various root scripts, as shown in section 6.1. We also found a trend between various web abstract tree nodes (like calls/local references/expression statements) and updates, reviews and comments.

The user network of TouchDevelop is very close to the pattern of Github (Social Coding). We base that on the fact that we could run a similar page rank based study on TouchDevelop as was done on Github[3].

The culture of updating scripts by different users and the community being strongly connected to each other via subscription also show us collaboration in a way we canachieve version control through different teams. This shows potential for Touchdevelop to have version control integrated in the app.

The Showcase scripts give us a better understanding of the application of TouchDevelop and that they are in sync with what TouchDevelop sought out to do.

There were quite a few challenges in this study, Collecting data from the script network was the biggest challenge.Understanding and running page rank algorithm on the user-user network. Visualizing TouchDevelop as more than what it presents itself presently, and studying the data to get to a more advanced version to itself.

VIII Future Work

In our work, we have studied the scripts and user network extensively. We couldn't do a very detailed study on the web AST of the scripts with respect to design, space and time complexity. With information about the node types, like calls, references, expression statements etc. This information could give us further insight on what these scripts are essentially built off.

We drew a very strong parallel between Social Coding and the developing environment of touchDevelop, this could further trigger features like in built version control console or more centralized way of looking at all the scripts and their updates.

With this study, we saw that TouchDevelop as claimed in [5],[6],[7] is more than just a training tool to create small mobile apps. It can be taken into consideration for actual product development at industry level in future.

We also, couldn't study art and features to a great extent since there was not enough information on the Rest APIs. It would be interesting to find out how diverse the use of art and features is on the app.

References

[1] Social Media for Software Engineering (2010) Begel. A, DeLine. R, Zimmermann. T FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research. Santa Fe, New Mexico, USA

[2] Model Comparison: A Key Challenge for Transformation Testing and Version Control in Model Driven Software Development (2004) Lin. Y, Zhang. Z, and Gray. J, Control in Model Driven Software Development. OOPSLA/GPCE: Best Practices for Model-Driven Software Development

[3] Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub Vasilescu. B, van Schuylenburg. S and Wulms. J, 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2014

[4] Creating a shared understanding of testing culture on a social coding site, Pham. R, Singer. L and Liskin. O, 2013 35th International Conference on Software Engineering (ICSE), 2013

[5] TouchDevelop — App Development on Mobile Devices, Tillmann. N, Moskal. M and Fahndrich. M, Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 2012

[6] An experiment in developing small mobile phone applications comparing on-phone to off-phone development, Nguyen. T, Rumee. S and Csallner. C, User Evaluation for Software Engineering Researchers (USER), 2012

[7] Microsoft Touch Develop and the BBC micro:bit, Ball. T, Bishop. J and Riley. C, IEEE/ACM 38th IEEE, ICSE '16 Proceedings of the 38th International Conference on Software Engineering Companion, Austin, Texas,2016

[8] Network Structure of Social Coding in GitHub, Thung. F, Bissyandé. T and Lo. D, 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013

[9] Towards Understanding Twitter Use in Software Engineering: Preliminary Findings, Ongoing Challenges and Future Questions, Bougie. G, Starke. J and German. D, Web2SE '11 Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering, 2011

[10] http://www.swtestacademy.com/jenkins-github-integration/, Nov 2, 2017

[11] S. Black, R. Harrison, and M. Baldwin. A survey of social media use in software systems development. In Proc. of the 1st Workshop on Web 2.0 for Software Engineering, ACM, 2010.

[12] TouchDevelop-backend/docs/Rest-APIs, 16 Oct-30-Oct,2017 (https://github.com/Microsoft/TouchDevelop-backend/blob/master/docs/REST-APIs.md)

[13] Using Version Control to Observe Student Software Development Process, Glassy. L, Journal of Computing Sciences in Colleges Volume 21 Issue 3, 2006

[14] Coordination in software development, Kraut. R and Streeter. L, Communications of the ACM Volume 38 Issue 3, 1995

[15] Linux.Com/Git 2.7.3 Out Now, Remains the Most Popular Source Code Management System Web,2016, softpedia, (https://www.linux.com/news/git-273-out-now-remains-most-popular-source-code-management-system)

[16] PageRank Citation Ranking: Bringing Order to Web, Page. L, Brin. S and Motwani. R, Proceedings of the 7th International World Wide Web Conference, 1998

[18] Nikolai Tillman,Sihan Li, Tao Xie , A Comprehensive Field Study of End-User Programming on Mobile Devices. 2013, Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium, San Jose, California

[19] Balaji Athreya, Faezeh Bahmani, Alex Diede, Chris Scaffidi , End-user programmers on the loose: A study of programming on the phone for the phone ,2011, Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on, Austria

[20] Akhin, M, Tillmann, N, Fahndrich, M, de Halleux, J, Moskal, M. (2011) *Code Similarity in TouchDevelop: Harnessing Clones*, Technical Report MSR-TR-2011-103, Microsoft Research.

[21] Anderson E, Li S, Xie T, (2013), A Preliminary Field Study of Game Programming on Mobile Devices, CoRR abs/1310.3308.