A COSMIC FUNCTION POINT TEST EFFORT ESTIMATION MODEL FOR MOBILE APPS: AN INDUSTRIAL CASE STUDY

Anureet Kaur[1*], Kulwant Kaur[2]

[1] *Asst Professor,Khalsa College,Amritsar,Punjab*
[2] *Asst. Professor, Apeejay Institute of Management Technical Campus Jalandhar, Punjab*[1]
anumahal@gmail.com, [2] kulwantkaur@apjimtc.org

Abstract: *Development and testing of mobile software are deemed distinctive from building traditional software encroaching to its distinctive features and thus entails extra efforts as paralleled to traditional software. Mobile application testing is an imperious and essential feat in the application development lifecycle ratifying quality and unwaveringly influences the development effort of the application. First objective of this paper is to validate the applicability of COSMIC test effort estimation model for mobile applications. Second objective is to obtain feedback from the software company for additional improvement in the model. A case study is conducted in a small size mobile software company. Results are reported via a comparative study after implementing the COSMIC test effort model in the form of a web tool. The results from the case study show that: (1) An informal expert-based estimation method used within the company is far less near to the actual effort incurred. (2) Mobile app and test factors act as a driving force in improving the test effort estimation. (3) Findings from the results determine the significance of mobile app factors which cannot be overlooked in the estimation progression of mobile software testing.*

Keywords: **mobile applications, test effort estimation, software engineering, case study**

1. INTRODUCTION

In contemporary years, progress in mobile technology has begotten an extortionate transformation in the day-to-day regime of human beings. Smartphones/mobile devices are proliferating in each demeanor of human life. There is not a single domain left where mobile technology has not assumed control. “A mobile app, short for mobile application or just app, is an application software designed to run on smart phones, tablet computers, and other mobile devices” Chen and Kotz, (2000). Mobile applications are available to be downloaded from app stores or are pre-installed by mobile manufactures. There are three types of mobile apps: Native, Web and Hybrid Apps.

- Native Apps: The applications developed for a particular platform and installed on the device e.g. they can be installed from the application store (Google Play, Apple app store, etc.).
- Web Apps: Web-based applications are available via the browser on device or third party browsers installed on the device.
- Hybrid Apps: These applications combine elements of both native and Web applications. They are available in the application store just like native apps and take the help of HTML to open in a browser like web apps.

The approach toward building up the mobile apps continues in the midst of the testing stage to confirm the accuracy of the mobile application. Test Estimation of mobile apps assist in lessening the comprised perils and thus succeeding with simple and accurate testing process.

There are various prevailing testing effort estimation techniques used for desktop/laptop software formed on judgment and rules of thumb, procedures based on analogy and work breakdown, practices based on factors and weights, techniques based on size, fuzzy and other models by Abhishek et al.,(2010), Arumugam and Babu (2015), Bhattacharyya and Malgazhdarov (2016), Islam et al., (2016), Jayakumar and Abran (2013), Nageswaran (2001), Sharma and Kushwaha, (2013), Souza and Barbosa (2010), Srivastava et al., (2012), Srivastava (2015), Srivastava et al. (2014), Zapata-Jaramillo and Torres-Ricaurte (2014). Certain authors Aranha and Borba (2007), Wadhwani et al. (2008) have suggested test effort estimation models for mobile apps and Parvez (2013) has altered traditional testing effort estimation techniques to mobile software. But none have measured characteristics/factors explicit to mobile applications Kaur and Kaur (2018). One model based on COSMIC function point is proposed by Kaur and Kaur (2019) for estimating the test effort in mobile software. The model is also validated using k-fold cross-validation method. The mobile app characteristics are also given due weight along with test factors.

The first objective of this research is to further validate the proposed model by Kaur and Kaur (2019) using a case study to identify its actual implementation in the real mobile software industry. An additional objective is to study the present method followed for test estimation in the mobile software industry and recognizing how practicing of proposed test effort model in form of a web tool affects the testing process. Also to recognize additional existent challenges encountered whilst test estimation of mobile apps.

The paper is divided into five sections. Section 2 summaries the COSMIC Test Effort Estimation Model proposed by authors in their previous work. Section 3 presents the design of research comprising information on company profile and data collection of the mobile app under consideration. Also, the computation of the test effort is presented in this section. The comparative result with the prevalent estimation technique and the proposed model is presented in section 4 along with some suggestions from the company professionals. Section 5 presents conclusion and future work.

2. COSMIC TEST EFFORT ESTIMATION MODEL: AN OVERVIEW

A novel test estimation model for mobile apps is proposed in Kaur and Kaur (2019) by contemplating the influence of different mobile app characteristics. The model implementation starts with felicitating mobile application requirements. From the requirements, the functional requirements are extricated and the size of functional requirements is computed using COSMIC function size measurement method. Details on COSMIC functional size method can be referred in in Kaur and Kaur (2019). The functional size is measured as a functional test size for input to the estimation model. The dataset of previously completed mobile applications is used to generate the regression test effort estimation model.

COSMIC FSM method is used to convert Functional User Requirements (FUR) into CFP. The manual CFP count is verified using a VisualFSM tool Pentad-SE (2014). The VisualFSM COSMIC Quick Start tool catered by Director of Pentad-SE Ltd can be used for computing CFP and is also accessible online for academic/industrial usage.

For figuring MobileFactor, 15 mobile app characteristics celebrated in Kaur and Kaur (2018) are measured. The formula for calculating MobileFactor is shown in equation (1).

$$MobileFactor = \sum_{i=1}^{15} MC_i * W_i \tag{1}$$

where MC_i is rating from 0 to 3 for i^{th} mobile characteristic and W_i is assigned a weight for the i^{th} mobile characteristic.

Similarly, for TestFactor calculation, seven test factors are measured. The formula for calculating TestFactor is shown in equation (2).

$$TestFactor = \sum_{i=1}^{7} TF_i * W_i \tag{2}$$

where TF_i is rating from 0 to 3 for i^{th} test factor and W_i is assigned a weight for i^{th} test factor.

3. RESEARCH DESIGN

3.1 Case Study Design

A case study is conducted following the guidelines in Runeson and Höst (2009). The goal of directing the case studies is to validate the applicability of the proposed model in real mobile apps test effort estimation. A case study is directed for mobile app test effort estimation on US-based software industry involved in development and testing of web and mobile applications. An execution of the case study followed the following steps:

1. Elicitation of the Function User Requirements (FURs) related to the mobile app to be developed and tested.
2. Estimation of the test effort using the test effort estimation model for mobile apps.
3. Evaluation of the results of estimation based on a comparative study with actual effort and prevalent estimation technique in the software company.

3.2 Company Context

The mobile application case study was conducted at SK TECHNOLOGIES, US. SK Technologies is a growing Web and Mobile App Development Company in the US with offices in India. The US site has around 50 employees. The mobile software department has 10 developers out of which 4 are under testing team and rest are fully devoted development teams. The documentation regarding user requirements provided by the developers was further converted to a format useful in the test effort estimation model.

3.3 Data Collection

3.3.1 Informal Interview*:* The purpose of the semi-structured interview with one manager, three developers, and two testers was to gain a better understanding of the estimation process prevalent in their company for mobile apps. One author asked the questions and other author noted down the answers. The questions in the interview were divided into following categories:

1. Related to the rating for Mobile app factors and Test factors.
2. Related to the present method followed for test estimation in the mobile software industry.
3. Related to knowing additional existent challenges encountered whilst test estimation of mobile apps.

The answer to question 1 is presented in the section 3.5 and 3.6. Section 4.1 presents the prevailing technique of testing estimation in the company. Section 4.2 presents challenges or feedback or limitations encountered in estimating test effort.

3.3.2 Documentation of Mobile Application Case Study (TradeInShop): The development team provided us with access to mobile app documentation starting from User requirements. The project was developed and tested under the guidance of the manager of the company. The development team involved two members and one tester. The actual effort for developing the project was 537 Person-Hour and took 230 Person-Hour for testing. The name of mobile app which is considered for case study is TradeInShop. TradeInShop mobile app allows ordering the items through a web shopping service shopping. So this app comes under category of mobile web app. The mobile app will be designed for Android mobile phones. The features of the app comprise of data handling, searching, selling items, adding them to favourites and allowing user messaging system. While running the app, users can access the features on the mobile phone and data available on a database can accessible with or without login. Table 1 provides description of the TradeInShop App.

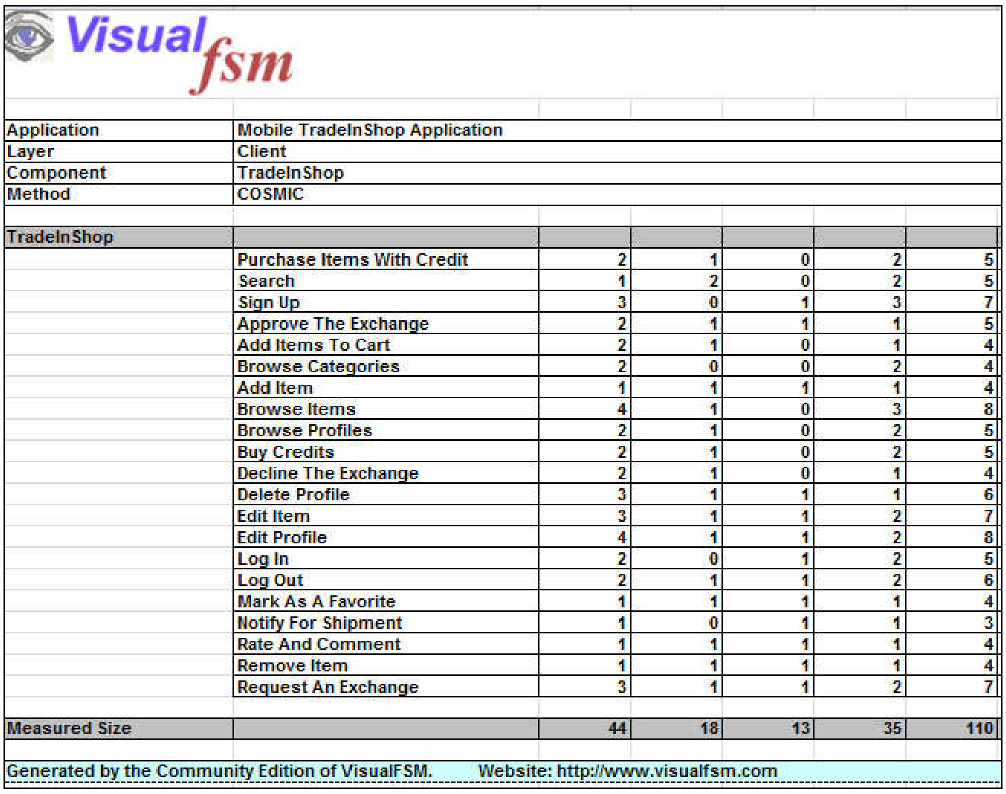
TABLE 1. DESCRIPTION OF MOBILE APP CASE STUDY FROM SK TECHNOLOGIES, US

Attributes	Description
Size of App	Medium
Type of Mobile App Domain	Shopping
Operating System	Android
Language Used	Java
Development Team	02 Developers
Testing Team	01 Tester
Actual Development Effort	537 Person-Hour
Actual Test Effort	230 Person-Hour
Test Tool Used	Robotium

Description of Functional User Requirements (FURs) for the TradeInShop mobile app is presented in Appendix A. These requirements will act as an input to the test effort estimation model for counting the functional size of the mobile app. The naming convention for each FUR is numbered from FUR1 to FUR21.

3.4 Counting COSMIC Function Points (CFP) for FURs using VisualFSM Tool

Figure 1 presents the summarized report for CFP count from VisualFSM tool. The CFP count obtained from VisualFSM tool acts as an input to a web tool used for calculating Test Effort for mobile apps. The web tool is developed using ASP.NET and Visual C# programming language. Figure 2 shows the input of CFP test size along with the mobile app name and description into the web tool for further calculation.

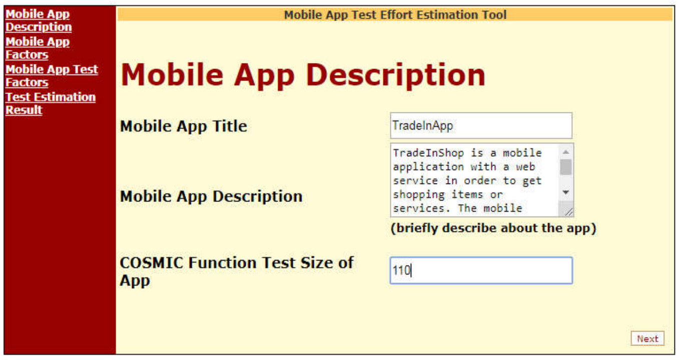


Application	Mobile TradeInShop Application
Layer	Client
Component	TradeInShop
Method	COSMIC

TradeInShop						
	Purchase Items With Credit	2	1	0	2	5
	Search	1	2	0	2	5
	Sign Up	3	0	1	3	7
	Approve The Exchange	2	1	1	1	5
	Add Items To Cart	2	1	0	1	4
	Browse Categories	2	0	0	2	4
	Add Item	1	1	1	1	4
	Browse Items	4	1	0	3	8
	Browse Profiles	2	1	0	2	5
	Buy Credits	2	1	0	2	5
	Decline The Exchange	2	1	0	1	4
	Delete Profile	3	1	1	1	6
	Edit Item	3	1	1	2	7
	Edit Profile	4	1	1	2	8
	Log In	2	0	1	2	5
	Log Out	2	1	1	2	6
	Mark As A Favorite	1	1	1	1	4
	Notify For Shipment	1	0	1	1	3
	Rate And Comment	1	1	1	1	4
	Remove Item	1	1	1	1	4
	Request An Exchange	3	1	1	2	7
Measured Size		44	18	13	35	110

Generated by the Community Edition of VisualFSM. Website: http://www.visualfsm.com

Figure 1. Final Reports with CFP Count



Mobile App Test Effort Estimation Tool

Mobile App Description

Mobile App Title: TradeInApp

Mobile App Description: TradeInShop is a mobile application with a web service in order to get shopping items or services. The mobile

(briefly describe about the app)

COSMIC Function Test Size of App: 110

Figure 2. Input to Mobile App Test Effort Estimation Tool

3.5 Non- Functional User Requirements Description (MobileFactor)

The identified mobile application characteristics are considered under non-functional user requirements. The ratings are assigned to each characteristic based on user requirements and collected through the interview from testers. The ratings for each mobile app characteristic are entered in a DropDownList on the web tool as shown in figure 3.

(1) Limited Memory (LimM): Test the app for memory requirement for installation and related files. So value 2 is assigned covering under the average impact.

(2) Limited CPU (LimC): Test the app that it should not consume more than 40 % of CPU while running on mobile devices. So value 1 is assigned for this factor.

(3) Limited RAM (LimR): Test the app if it consumes more than 45% of RAM while running. So this requires an average test with value 2 assigned for 40% to 70% RAM consumption.

(4) Limited screen size (LimSS): The app should be tested for content display on varying screen size of Samsung devices and the app outlook should not vary from its original design at most 60% and also test for both portrait and landscape mode and also for different orientation (portrait and landscape). So value 2 is assigned for this constraint.

(5) The diversity of User interfaces (touchscreen, keypad, voice) (UserInter): The app should be tested at minimum for its behavior when input is through touchscreen and keypad. So value 2 is assigned.

(6) Context-awareness (CAwar): Not applicable. So test is not performed with 0 value assigned.

(7) Diverse Mobile Connections (2G, 3G, 4G, and various wireless networks (MConn). This app needs to be connected to internet for accessing web service, so it should be tested for at least 2G,3G and over Wi-Fi networks. So this is a major factor and value 3 is assigned.

(8) Different application types (Native, Hybrid, Web): As this is a mobile web app, so value 2 is assigned for this factor.

(9) Diverse operating systems (software) (DivOS): The app is being developed and tested only for the android platform. So only testing for android OS is involved giving value 1for this factor.

(10) Diverse devices (hardware) (DivH/w): The app should be tested for Samsung mobile devices in general. So with this requirement value, 1 is assigned for this factor.

(11) Interrupt (Int): The app should handle the incoming calls, comprising under category of 40% to 70%. So value 2 is assigned for this factor.

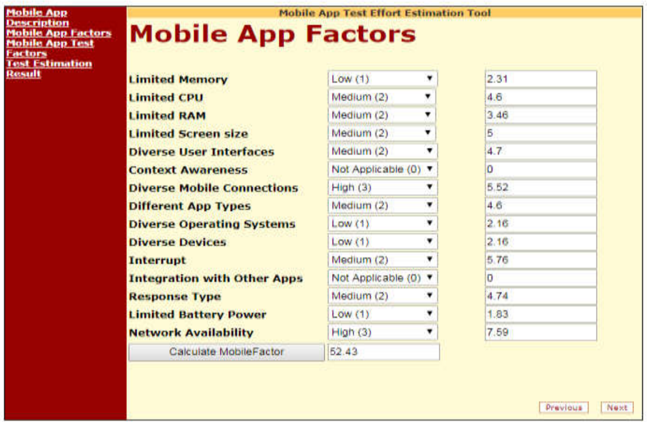
(12) Integration with other Apps (IOA): The app will not be integrated with other installed app on the mobile. So testing won't be required for this aspect. So with N.A., value 0 is assigned.

(13) Response Time (RT): The app should be tested for its response time i.e. 2 seconds at most. So value 2 is assigned as it comes under the average testing requirement.

(14) Limited Battery power (LBT): The app should be tested when the battery power reaches below 10%. As this is following some standards by given platform to preserve battery, value 1 is assigned.

(15) Network Availability (NetAv): As this app needs to be connected to internet for accessing web service, so it should be tested for network connectivity availability and its behavior with poor or no connection. So this is a major factor and value 3 is assigned.

After pressing the *"Calculate MobileFactor"* button on the web tool, the value of MobileFactor is displayed. Then press the next button to move to the next page.



Mobile App Factors

Limited Memory	Low (1)	2.31
Limited CPU	Medium (2)	4.6
Limited RAM	Medium (2)	3.46
Limited Screen size	Medium (2)	5
Diverse User Interfaces	Medium (2)	4.7
Context Awareness	Not Applicable (0)	0
Diverse Mobile Connections	High (3)	5.52
Different App Types	Medium (2)	4.6
Diverse Operating Systems	Low (1)	2.16
Diverse Devices	Low (1)	2.16
Interrupt	Medium (2)	5.76
Integration with Other Apps	Not Applicable (0)	0
Response Type	Medium (2)	4.74
Limited Battery Power	Low (1)	1.83
Network Availability	High (3)	7.59
Calculate MobileFactor	52.43	

Figure 3. MobileFactor Calculation

3.6 Test Characteristics of App (Testfactor Consideration)

The test characteristics identified from (2013), de Almeida et al., (2009,2001,2014). These test characteristics are rated collected through the interview from testers and testing requirements. Again the ratings for each test characteristic are entered in a DropDownList on the web tool as shown in figure 4.

(1) Test Tools (TT): The testing tool used by the testing team is robotium helping to automate the test process. So testing complexity is expected to decrease as compared to manual testing. The value 1 is assigned due to this reason.

(2) Documented Inputs (DocInp): The SRS documents are used for test input with good quality. So value 0 is assigned as it may not add to testing complexity.

(3) Development Environment (DevEnv): The developers used eclipse environment with average development resource. So value 2 is assigned to this test factor.

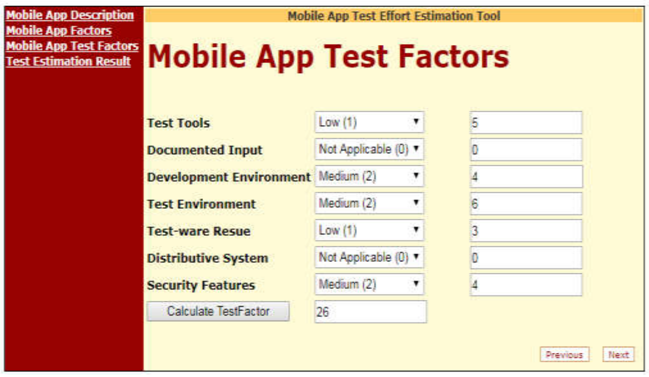
(4) Test Environment (TEnv): The app will be tested on emulators of the hardware as the availability of all mobile devices running Android OS is not available with the testing team. Only three Samsung mobile devices are used to perform real device testing. Value 2 is assigned for this factor.

(5) Test-ware Reuse (TR): The testing tools, test scripts are reused decreasing the test redesigning and further decreasing the test complexity. So value 1 is assigned.

(6) Distributive System (DSys): No distributive environment. So with N.A. value 0 is assigned.

(7) Security Features (SecFea): As login features with username and password are used for accession with web server so average security is required to protect its information and data. Value 2 is assigned for this constraint.

Again the ratings for each test characteristic are entered in a DropDownList on the web tool as shown in figure 4. After pressing Calculate TestFactor button on the web tool, the value of TestFactor is displayed. Then Press the next button to move to the next page.



Mobile App Test Factors

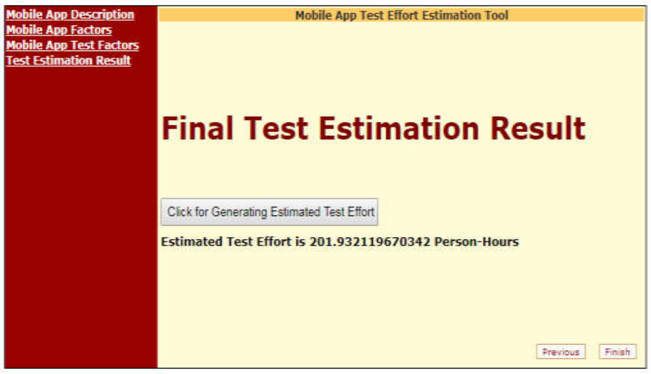
Test Tools	Low (1)	5
Documented Input	Not Applicable (0)	0
Development Environment	Medium (2)	4
Test Environment	Medium (2)	6
Test-ware Resue	Low (1)	3
Distributive System	Not Applicable (0)	0
Security Features	Medium (2)	4
Calculate TestFactor	26	

Figure 4. TestFactor Calculation

3.7 Test Effort Estimates with Proposed Model (Kaur and Kaur, 2019a)

The proposed model is quite simple to use and its applicability is presented with a mobile project case study. After gathering the inputs required for the proposed model with CFP=110, MobileFactor= 52.44 and TestFactor= 26. The inputs are fed into the MLR equation (3) in Kaur and Kaur (2019) and implemented as a web tool shown in figure 5.

$$TestEffort = 0.309 * (CFP^{1.103} + MobileFactor^{0.164} + TestFactor^{0.199}) \quad (3)$$



Final Test Estimation Result

Estimated Test Effort is 201.932119670342 Person-Hours

Figure 5. Final Report on test estimation of TradeInShop

The web tool yielded an estimate of 207.655Person-Hour which is required to test the app. Further the estimated effort can be used for setting the cost incurred on testing. If the tester is paid $5 per hour for app testing then $1025 can be quoted as testing cost.

3.8 Accuracy Parameters

The assessment measure to calculate the accuracy of the estimate is MRE which is exercised in this paper. *MRE (Magnitude of Relative Error*): MRE is a prevalent measure to evaluate estimation models (2012). The formula for MRE calculation is presented in equation (4)

$$MRE = \frac{|\,ActualTestEffort - \Pr edictedTestEffort\,|}{ActualTestEffort} \qquad (4)$$

4. RESULTS FROM CASE STUDY

4.1 Comparison of Test Effort Estimation Results

In this case study, different experiments are performed to compare the proposed methodology against a traditional approach where the test effort estimation is generated by a human expert and is asked to estimate the new project's effort based on personal experience and knowledge. This informal expert-based estimation method used within the organization is far less near to the actual effort incurred. Expert judgment effort estimation techniques are based on the person's experience and intuition. The evaluation indices MRE is calculated for the proposed model and expert estimation method prevalent in SK Technologies indicates that the proposed model performs better than expert estimation as shown in table 2. The proposed model gives better MRE with 9.71% as compared to MRE for expert estimation i.e. 30.4%. Figure 6 shows that the proposed test effort estimation model predictions are more near to actual test effort. Figure 7 graphically shows that MRE percentage is high with an expert estimation as compared to the proposed model.

TABLE 2. COMPARISON OF PREVALENT TEST EFFORT ESTIMATION TECHNIQUE WITH PROPOSED MODEL

Parameters	TradeInShop Mobile App
Actual Test Effort(P-H)	230
Estimated Test Effort by Proposed Model	207.655
Estimated Test Effort by Expert Estimation	300
MRE Proposed Model	9.71%
MRE Expert Estimation	30.4%

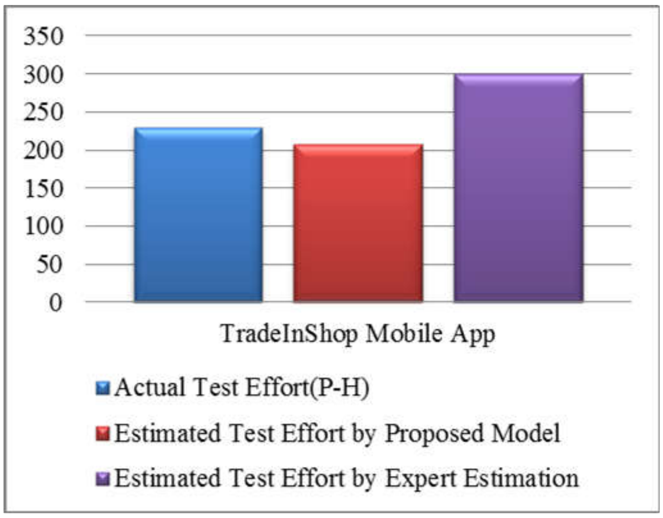
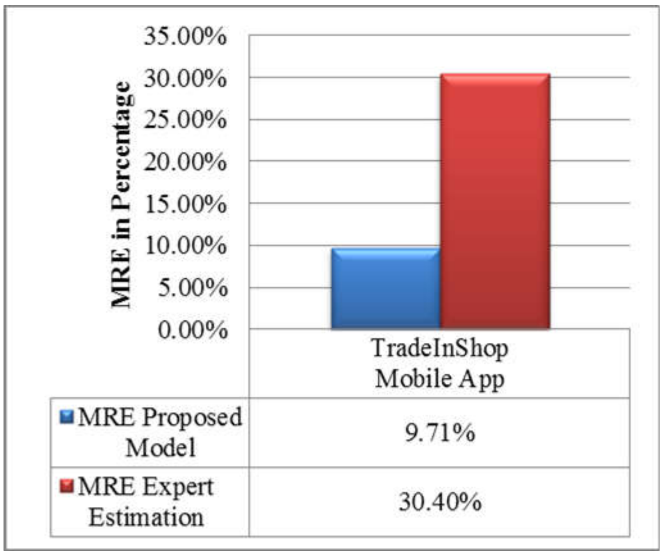


Figure 6. Comparison of Proposed Model with Expert Estimation Technique



	TradeInShop Mobile App
🟦 MRE Proposed Model	9.71%
🟥 MRE Expert Estimation	30.40%

Figure 7. MRE Comparison for Proposed model and Expert Estimation

4.2 Feedback on estimating the mobile application testing process using the proposed model

The interview answers regarding feedback on estimating the mobile application testing process using the proposed model are listed below:

- One tester suggested introducing more factors while estimating the test effort such as test team size and tester experience.
- Also, another tester suggested considering test cycle phase-wise prediction e.g. Test execution effort estimation can be explored after the mobile app is developed and before testing is started.
- Testers think that there might be more than 15 mobile app characteristics that need consideration while estimation, i.e. security testing of the app, data integrity, etc.
- Needs documentation and little experience in using VisualFSM tool for counting COSMIC Function Point (CFP).
- The web tool implementing the proposed test effort estimation is very easy to use, but the assignment of ratings for MobileFactor and Testfactor always requires help in form of handouts.
- One tester asked about the model basis (Regression based Model) and suggested to use other machine learning techniques for prediction to check if the accuracy of the prediction further improves or not.

5. CONCLUSION AND FUTURE REMARKS

In this paper, a case study is presented that shows usage of the proposed test effort estimation model in real software companies. First, the company profile of the case study is presented, followed by a description of the mobile application software to be developed and tested. The Functional User Requirements (FURs) of mobile apps are presented. The VisualFSM tool is used to calculate the functional or test size of mobile apps from FURs. The test effort estimation tool for mobile apps is developed based on the proposed model by authors. This tool is developed using ASP.NET and C# programming language. The functional or test size in terms of CFPs is given as an input to the tool. To calculate MobileFactor and TestFactor, ratings for mobile app characteristics and test characteristics are collected in the range from 0 to 3 in a DropDownList of the tool. The final screen in the tool presents the final test effort in terms of Person-Hour. The results obtained using the tool is compared with the prevalent estimation technique in the company. The results clearly show that the proposed model gives a better prediction as compared to expert estimation currently used in the company. One of the limitations of the proposed work is the scarcity of mobile app projects available to create the dataset. Published datasets such as ISBSG (2007), NASA (2007), PROMISE (2004), Experience (2007), and Maxwell (2000) do not include information about mobile apps. The proposed test effort model which focused on the mobile apps has been validated and also measured for accuracy but it is considered that the proposed model needs to be improved for future needs. The following recommendations in form of interview are collected to improve the model as future work.

1. Machine learning techniques and case-based reasoning can be exploited to predict test effort and compare the results obtained with the proposed Regression model.

2. Mobile app specific characteristics identified in the literature are mapped to NFR glossary in Kaur and Kaur (2018). There can be other characteristics not identified in the literature survey which can further be explored and mapped to NFR that may affect the test effort.

3. Test factors such as test team size and tester experience can be considered for estimating test effort.

4. Test cycle phase-wise prediction e.g. Test execution effort estimation can be explored after the mobile app is developed and before testing is started.

APPENDIX A: Functional User Requirements for TradeInShop App

(1) FUR1: Sign Up

The User enters the User details comprising a username, name, surname, password, e-mail address and link to profile Photo.

x- Trader Fills the signup message comprising the user details then sends it to the Web Service which response by sending an e-Mail with an activation code to the User.

The User enters the activation code then x-Trader sends it to the Web Service

If the sign-up succeeds then x-Trader receives a confirmation message from Web Service then displays it to the user.

x-Trader creates the User Profile in the local database

If the sign-up fails then x-Trader receives an error message from Web Service describing the error then displays it to the user.

(2) FUR2: Log In

The User enters the user credentials comprising username or e-Mail and the password.

x-Trader sends the logon request credentials to the Web Service.

If the logon succeeds x-Trader receives a logon success message from the Web Service which it displays to the User.

x-Trader updates the User logon state in the local database.

If the logon fails x-Trader receives an error message describing the error from the Web Service which it displays to the User.

(3) FUR3: Log Out

The User enters a request to logout.

x-Trader retrieves the User Profile credentials from the local database

x-Trader constructs the logout message comprising the credentials then sends them to the Web Service.

If the logout succeeds then x-Trader receives a logout success from the Web Service then displays it to the User.

x-Trader updates the Logon state in the local database.

If the logout fails then x-Trader receives an error message from the Web Service then display it to the User,

(4) FUR4: Search

The User enters the search criteria comprising item price, location, or category.

x-Trader constructs then search items message comprising the search criteria then x-Trader sends it to the Web Service

x-Trader receives the Item list from the Web Service then displays it to the user.

(5) FUR5: Browse Profiles

The user enters a request for information about all Profiles.

x-Trader gets the User credentials from local database

x-Trader constructs the browse profiles massage comprising the request ID as the credentials then sends it to the Web Service,

x-Trader receives a list of users comprising the name, profile, photo and a list of items posted.

x-Trader displays the list of users and items lists to the user.

(6) FUR6: Browse Categories

The user enters a request for information about all categories.

x-Trader gets the user credentials from the local database.

x-Trader constructs the browse Categories message comprising the request ID and the credentials then sends it to the Web Service,

x-Trader receives a list of Categories comprising the name and the number of items in the category from the Web Service.

x-Trader displays the Category list to the User.

(7) FUR7: Browse Items

The user enters a request for information about all Items.

x-Trader gets the User credentials from the database.

x-Trader constructs the browse items message comprising the request ID and credentials then sends it to the Web Service,

If the request succeeds x-Trader receives a list of Items comprising the name, image, prices, and features from the Web Service then displays it to the User.

If the request fails then x-Trader receives an error message describing the problem from the Web service then displays it to the user.

(8) FUR8: Edit Profile

The User enters the details of the User to be edited.

x-Trader gets the User credentials from the local database.

x-Trader constructs the edit profile message comprising credentials and User Details.

x-Trader sends the edit Profile message to the Web Service. The web service responds by sending an e-Mail comprising a confirmation code to the User.

The User enters the confirmation code then x-Trader sends it to the Web Service.

If the edit succeeds then x-Trader receives a success message from the Web Service and displays it to the User.

x-Trader updates the User profile in the local database.

If the edit fails then X-Trader receives an error message describing the error from the Web Service and displays it to the User.

(9) FUR9: Buy Credits

The User enters the no of credits to apply to the account.

x-Trader gets the user credentials from the database.

x-Trader constructs the add credit message comprising no credits and credentials then sends it to the Web Service.

The Web Service redirects the user to the Pay-Pal portal where payment is made.

If the payment is successful x-Trader receives a confirmation message from the Web service

x-Trader displays it to the user.

(10) FUR10: Add Items to Cart.

The User enters the Item ID.

x-Trader gets the user credentials from the database.

x-Trader constructs the add item message comprising the Item ID and credentials then sends it to the Web Service.

If the add succeeds x-Trader receives a confirmation message from the Web Service then displays it to the User.

(11) FUR11: Purchase Items with Credit

The User enters the list of items to be purchased comprising the item IDs.

x-Trader gets the user credentials from the database.

x-Trader constructs the purchase message comprising the item list and credentials then sends it to the Web Service.

If the purchase succeeds x-Trader receives a message from the Web Service containing the number of credits remaining which is displayed to the User

If the purchase fails x-Trader receives an error message describing the problem from the Web Service which it displays to the User.

(12) FUR12: Request an Exchange

The User enters the offered Item ID and the required Item ID.

x-Trader gets the user credentials from the database

x--Trader constructs the request exchange message comprising offered item id, required item id, and credentials then send it to Web Services.

if the request succeeds then x-Trader receives a success message from the Web Service.

x-Trader updates the Exchange state in the local database.

if the request fails then x-Trader receives an Error message from the Web Service which it displays to the User.

(13) FUR13: Approve the Exchange

x-Trader receives a request approve message comprising the offered Item ID, the required Item

ID and the seller profile from the Web Service.

The User enters the exchange approve indicator.

x-Trader gets the user credentials from the local database.

x-Trader constructs the approve exchange message comprising the original message contents and the approve indicator then sends it to the Web Service.

x-Trader updates the Item exchange state in the local database.

(14) FUR14: Decline the Exchange

x-Trader receives a request approve message comprising the offered Item ID, the required Item ID and the seller profile from the Web Service.

The User enters the request decline indicator.

x-Trader gets the user credentials from the local database.

s-Trader constructs the decline exchange message comprising the request Id and credentials then sends it to the Web Service.

(15) FUR15: Notify For Shipment

The User enters the email message comprising Item Details, shipping date, and buyer e-mail.

x-Trader sends the e-Mail message to the User

x-Trader updates the exchange item state in the local database.

(16) FUR16: Add Item

The User enters the Item List comprising for each item details, images, amount and price.

x-Trader gets the user credentials from the local database.

x-Trader constructs the approve exchange message comprising the item list and credentials then sends it to the Web Service

x-Trader updates the Item exchange state in the local database.

(17) FUR17: Edit Item

The user enters the details of the item to be edited.

x-Trader gets the user credentials from the Database.

x-Trader constructs the edit item message comprising item details and credentials, then sends it to the Web Service

If the edit succeeds x-Trader receives a success message from the Web Service.

x-Trader updates the Item details in the local database.

If the edit fails x-Trader receives an error message from the Web Service which it displays to the User.

(18) FUR18: Remove Item

The user enters the item ID.

x-Trader gets the user credentials from the Database

x-Trader constructs the remove item message comprising the item ID and credentials, then sends it to the Web Service

If the delete succeeds x-Trader deletes the item from the local database.

(19) FUR19: Delete Profile

The User enters the ID of the user to be deleted.

x-Trader gets the user credentials from the Database x-Trader

x-Trader constructs the delete profile message comprising deleted user ID and credentials.

x-Trader sends the delete profile message to the Web Service if the delete succeeds x-Trader receives a success message from the Web Service

x-Trader deletes the profile from the local database.

If the delete fails x-Trader receives an error message from the Web Service which is display to the User.

(20) FUR20: Mark as a Favourite

The user enters the item ID.

x-Trader gets the user credentials from the Database.

x-Trader constructs the mark favourite message comprising the item Id and credentials then sends it to the Web Service

x-Trader updates the Item in the local database.

(21) FUR21: Rate and Comment

The user enters the rating details comprising item ID, comments and rating.

x-Trader gets the user credentials from the Database.

s-Trader constructs the rating message comprising the rating details and credentials then sends it to the Web Service.

x-Trader updates the User rating and comments in the local database.

REFERENCES

[1] Abhilasha, Sharma, A., 2013. Test effort estimation in regression testing, in: Innovation and Technology in Education (MITE), 2013 IEEE International Conference in MOOC. pp. 343–348.

[2] Abhishek, C., Kumar, V.P., Vitta, H., Srivastava, P.R., 2010. Test Effort Estimation Using Neural Network. J. Softw. Eng. Appl. 03, 331–340.

[3] Aranha, E., Borba, P., 2007. An Estimation Model for Test Execution Effort, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). IEEE, pp. 107–116.

[4] Arumugam, C., Babu, C., 2015. Test Size Estimation for Object Oriented Software Based on Analysis Model. J. Softw. 10, 713–729..

[5] Bhattacharyya, A., Malgazhdarov, T., 2016. PredSym: estimating software testing budget for a bug-free release. Proc. 7th Int. Work. Autom. Test Case Des. Sel. Eval. - A-TEST 2016 16–22.

[6] Chen, G., Kotz, D., 2000. A Survey of Context-Aware Mobile Computing Research. Dartmouth Comput. Sci. Tech. Rep.

[7] de Almeida, É.R.C., de Abreu, B.T., Moraes, R., 2009. An Alternative Approach to Test Effort Estimation Based on Use Cases, in: 2009 International Conference on Software Testing Verification and Validation. IEEE, pp. 279–288.

[8] Experience, 2007. Experience datasets [WWW Document]. URL http://www.fisma.fi.

[9] ISBSG, 2007. ISBSG Dataset [WWW Document]. URL https://www.isbsg.org/project-data/

[10] Islam, S., Pathik, B.B., Khan, M.H., Habib, M., 2016. Software test estimation tool: Comparable with COCOMOII model, in: IEEE International Conference on Industrial Engineering and Engineering Management. pp. 204–208.

[11] Jayakumar, K.R., Abran, A., 2013. A Survey of Software Test Estimation Techniques. J. Softw. Eng. Appl. 6, 47–52.

[12] Kaur, A., Kaur, K., 2019a. A COSMIC function points based test effort estimation model for mobile applications. J. King Saud Univ. - Comput. Inf. Sci.

[13] Kaur, A., Kaur, K., 2019b. Investigation on test effort estimation of mobile applications: Systematic literature review and survey. Information Software. Technolology Volume 110, June 2019, Pages 56-77.

[14] Kaur, A., Kaur, K., 2018. Systematic Literature Review of Mobile Application Development and Testing Effort Estimation. J. King Saud Univ. - Comput. Inf. Sci.

[15] Maxwell, 2000. Maxwell datasets [WWW Document]. URL http://www.promisedata.org/?p=108.

[16] Nageswaran, S., 2001. Test Effort Estimation Using Use Case Points, in: In: Proceedings of 14th International Internet Software Quality Week.

[17] NASA, 2007. NASA datasets [WWW Document]. URL http://data.giss.nasa.gov/.

[18] Nassif, A.B., Capretz, L.F., Ho, D., 2012. Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model, in: 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. IEEE, pp. 589–594.

[19] Parvez, A.W.M.M., 2013. Efficiency factor and risk factor based user case point test effort estimation model compatible with agile software development, in: Proceedings - 2013 International Conference on Information Technology and Electrical Engineering: Intelligent and Green Technologies for Sustainable Development", ICITEE 2013. Yogyakarta, Indonesia., pp. 113–118.

[20] Pentad-SE, 2014. VisualFsm The Software Measurement Tool [WWW Document]. URL http://www.visualfsm.com/ (accessed 12.3.18).

[21] PROMISE, 2004. PROMISE datasets [WWW Document]. URL http://www.promisedata.org

[22] Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng.

[23] Sharma, A., Kushwaha, D.S., 2013. An empirical approach for early estimation of software testing effort using SRS document. CSI Trans. ICT 1, 51–66.

[24] Souza, P.P., Barbosa, M.W., 2010. Tailoring the Test Point Analysis Estimation Technique in a Software Testing Process, in: IV Encontro Brasileiro de Testes (EBTS) At: Recife.

[25] Srivastava, P.R., 2015. Estimation of software testing effort using fuzzy multiple linear regression. Int. J. Softw. Eng. Technol. Appl. 1, 145.

[26] Srivastava, P.R., Bidwai, A., Khan, A., Rathore, K., Sharma, R., Yang, X.S., 2014. An empirical study of test effort estimation based on bat algorithm. Int. J. Bio-Inspired Comput. 6, 57.

[27] Srivastava, P.R., Varshney, A., Nama, P., Yang, X.S., 2012. Software test effort estimation: a model based on cuckoo search. Int. J. Bio-Inspired Comput. 4, 278.

[28] Wadhwani, V., Memon, F., Hameed, M.M., 2008. Architecture based reliability and testing estimation for mobile applications, in: Communications in Computer and Information Science. Springer, Berlin, Heidelberg, pp. 64–75.

[29] Zapata-Jaramillo, C.M., Torres-Ricaurte, D.M., 2014. Test Effort: a Pre-Conceptual-Schema-Based Representation. Dyna 81, 132–137.