Journal of King Saud University – Computer and Information Sciences 34 (2022) 1–15

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Systematic literature review of mobile application development and testing effort estimation

Anureet Kaur [a,*], Kulwant Kaur [b]

[a] I.K. Gujral Punjab Technical University, Kapurthala, India
[b] School of IT, Apeejay Institute of Management Technical Campus Jalandhar, India

ARTICLE INFO

Article history:
Received 11 June 2018
Revised 26 September 2018
Accepted 2 November 2018
Available online 3 November 2018

Keywords:
Mobile Applications
Estimation
Test effort
Systematic literature review
Agile

ABSTRACT

In the recent years, the advances in mobile technology have brought an exorbitant change in daily life-style of individuals. Smartphones/mobile devices are rampant in all aspects of human life. This has led to an extreme demand for developing software that runs on mobile devices. The developers have to keep up with this high demand and deliver high-quality app on time and within budget. For this, estimation of development and testing of apps play a pivotal role. In this paper, a Systematic Literature Review (SLR) is conducted to highlight development and testing estimation process for software/application. The goal of the present literature survey is to identify and compare existing test estimation techniques for traditional software (desktop/laptop) and for mobile software/application. The characteristics that make mobile software/application different from traditional software are identified in this literature survey. Further, the trend for developing the software is towards agile, thus this study also presents and compares estimation techniques used in agile software development for mobile applications. The analysis of literature review suggests filling a research gap to present formal models for estimating mobile application considering specific characteristics of mobile software.

Contents

* Corresponding author at: 60, Kabir Park,opp. Guru Nanak Dev University, Amritsar, Punjab, India.
E-mail addresses: anumahal@gmail.com (A. Kaur), kulwantkaur@apjimtc.org (K. Kaur).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2018.11.002
1319-1578/

1. Introduction

The mobile devices being utilitarian, user-friendly, accessible has made it the most popular and indispensable expedient for human essentials from the past few years (Malavolta et al., 2015). Mobile software developers' are driven to release software on time and within budget. Software estimation plays a pivotal role in providing the most accurate sizing figure for building confidence in developers and stakeholders relationship (Soares and Fagundes, 2017). Many approaches used for estimation of traditional software are adapted for mobile application development and testing (Wasserman, 2010).

The testing phase of traditional software development proceeds through additional life cycle called Software Testing Life Cycle (STLC) (Katherine and Alagarsamy, 2012). According to Gao et al. (2014) mobile software testing are set of activities for mobile apps on mobile devices by exhausting definite software test techniques and tools in order to confirm quality in functionality, performance, and QoS, as well as features, like mobility, usability, interoperability, connectivity, security and privacy. The main phases of the testing process include test planning, test designing, test execution and test analysis (Farooq et al., 2011; Amen et al., 2015).

The estimation of effort for software testing comprises an estimation of test size, effort (Person per Hour), cost and entire schedule by means of several methods, tools and techniques (Abhilasha and Sharma, 2013). If effort, time and cost required to test the software can be anticipated, the testing resources can be systematically planned within a set target date to ensure lucrative culmination of projects. According to Zhu et al. (2008b), for estimating the test effort the major consideration is given on test designing (creation of test cases) and test execution.

With the advent of Agile Software Development (ASD) (Usman et al., 2014) entire software development community has been driven by the adoption of agile methodology. The Agile approach to mobile application development states an iterative and incremental approach comprising self-organizing teams and cross-functioning teams working together to build the software (Kaur, 2016). The prominent existing agile mobile application development approaches are MOBILE-D, RaPiD7, Hybrid methodology, MASAM, Scrum with Lean Six Sigma (SLeSS) (Dewi and Nur Atiqah Sia, 2015). The Agile espousal to mobile application development is considered as a natural fit by many researchers (Cunha et al., 2011; Rahimian and Ramsin, 2008; Scharff and Verma, 2010). In an agile environment, development and testing are not considered separate phases as in traditional software development (Rahimian and Ramsin, 2008). The estimation of software in agile is prepared for both development and testing together. Estimation of effort in agile development is a new area of focus and very less work is reported literature (Aslam et al., 2017).

The significant contribution of the paper lies in examining the test effort estimation techniques for desktop/laptop software development and mobile software development. Further, the development and test effort estimation techniques are evaluated from two approaches of mobile application development process i.e., traditional software development and agile software development. Another major contribution is identifying the characteristics of mobile apps that make them distinct from traditional software.

Subsequently, the paper is divided as follows: Section 2 presents the research method comprising three phases of Systematic Literature Review (SLR). First and second phase is devoted to forming Research Questions (RQ) and finding relevant literature for studies. The results of the review are analyzed in the third phase i.e. result reporting phase of SLR, answering each Research Question (RQs). In section 3, discussions, research gaps and future directions are presented. Some threats to the validity of SLR are discussed in section 4 followed by conclusions in Section 5.

2. Research method

This section outlines the related literature and findings by the researchers which form the desired background for this research. The guidelines provided by Kitchenham and Charters (2007) are followed by conducting Systematic Literature Review (SLR). SLR is a research manner for carrying out a literature review in an orderly way of charting definite phases. SLR method uses three phases for performing literature review including Planning and specifying research questions, conducting the review that comprises an identification of search string and data sources, selecting studies, quality assessment, and data extraction and finally reporting the review. The steps followed for systematic literature review are undertaken in the following sections of this paper. The overview of the systematic literature review is shown in Fig. 1.

2.1. Planning phase

For the smooth conduct of systematic literature review, proper planning is fundamental for smooth execution of SLR. The research questions derive from the entire systematic literature review planning phase.

Affirming the research questions is the vital part of any systematic review. In accordance with guidelines proposed by Petticrew and Roberts (2006) the criteria to frame research questions are based on PICOC (Population, Intervention, Comparison, Outcomes, and Context). If the research question is not outlined properly, the literature review may turn out off course. For this study, PICOC is defined as shown in Table 1.

2.1.1. Research questions (RQs)

The review questions steer the entire systematic review methodology. The foremost aim of the review is to answer the following research question:-

RQ1. What are currently known software test estimation techniques for traditional applications?

RQ2. What are mobile development and testing estimation techniques?

This RQ can be subdivided into two sub-categories:-

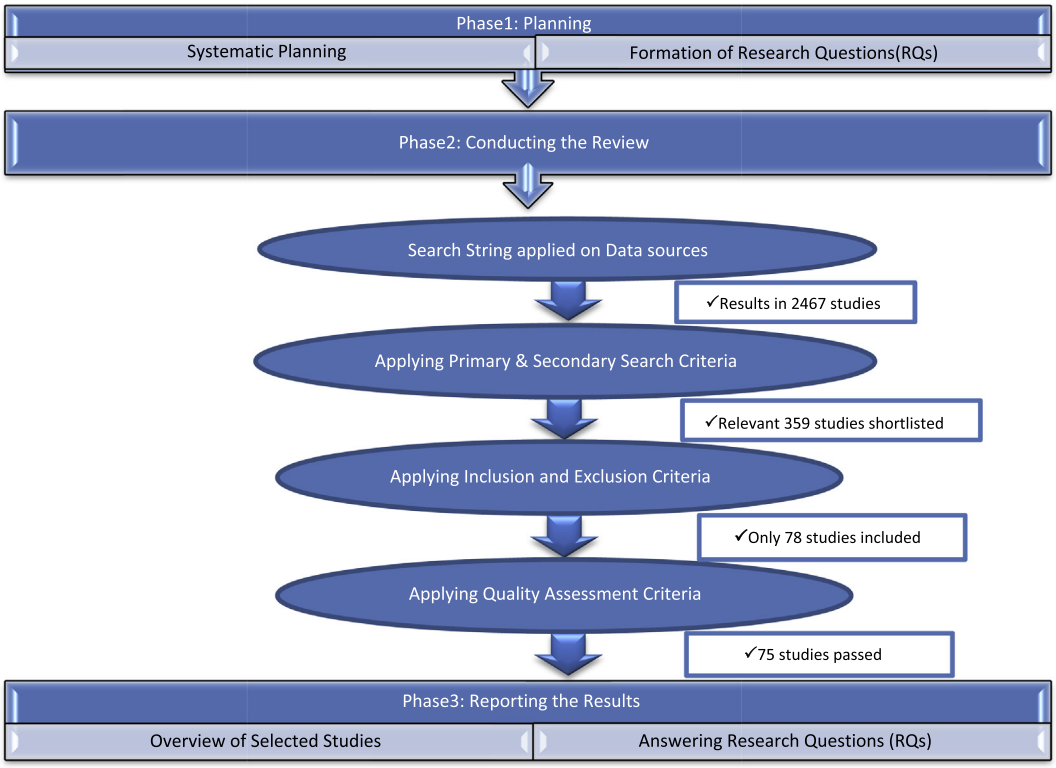

Fig. 1. Systematic Literature Review (SLR) Phases by Kitchenham and Charters (2007).

Table 1
PICOC with description.

PICOC	Description
Population	Mobile Application projects.
Intervention	Test Effort estimation techniques/methods/process.
Comparison	Traditional software test effort estimation techniques with mobile apps testing estimation.
Outcomes	Mobile software test estimation techniques and characteristics of mobile apps that are considered important in development and testing estimation.
Context	Review the existing studies on test estimation of mobile Apps.

RQ2.a. What are mobile development and testing estimation techniques in a traditional software development environment?
RQ2.b. What are mobile development and testing estimation Techniques in agile software development?

RQ3. How is the development and testing of mobile applications different from traditional software and what is different mobile app testing characteristics for estimation?

2.2. Conducting the review phase

2.2.1. Search strategy

The intent of the search strategy is to discover the studies that would assist in answering the RQs. The three phases of the search strategy comprise of identifying keywords and defining search strings, data sources selection and finally search process in data sources.

Table 2
List of keywords and synonyms.

Keywords	Synonymous Terms
Software	Software, project, system, application
Testing	Test, verification, validation
Effort	cost, resource, size, metric,
Estimation	Estimating, estimate, prediction, predicting, predict, assessment, forecasting, forecast, calculation, calculate, calculating, sizing, measure, measuring
Mobile Application	Mobile software, Mobile Apps, Mobile project
Development	Improvement, Progress
Method	Process, techniques, models, approaches
Agile	Scrum, XP, lean, crystal
Characteristics	Features, attribute, factors

2.2.1.1. Identifying keywords and defining search strings. The foremost phase of the search strategy is to ascertain the search string. The search strategy is set up to describe search strings and primary data sources. The guidelines provided by Kitchenham and Charters (2007) were followed to define the search string by analyzing the main keywords in RQs, synonyms of the keywords and on any other spellings of the words. Following are the identified keywords and synonyms are shown in Table 2:

Based on the identified keywords, the search string was obtained by joining synonymous terms using the 'OR', other keywords using logical 'AND' and wildcard character (*). Here wildcard character represents 0, 1, or any number of alphanumeric characters. The search string is categorized in four ways according to the RQs formed. Table 3 lists the categories and corresponding search string.

Table 3
Search string categories.

Category	Based on RQs	Search string
1	Software Testing effort Estimation Techniques (RQ1)	(“software” OR “project” OR “system” OR “application”) AND (“Test*” OR “verification” OR “validation”) AND (“Effort” OR “cost” OR “resource” OR “size” OR “metric”) AND (“estimate*” OR “predict*” OR “assessment” OR “forecast*” OR “calculate*” OR “sizing” OR “ measure*”) AND (“Process” OR “techniques” OR “models” OR “ approaches”)
2	Mobile Application Development and Testing effort Estimation Techniques (RQ2.a.)	(“Mobile Application” OR “Mobile software” OR “ Mobile App” OR “ Mobile project”) AND (“Develop*” AND “Test*” OR “verification” OR “validation”) AND (“Effort” OR “cost” OR “resource” OR “size” OR “metric”) AND (“estimate*” OR “predict*” OR “assessment” OR “forecast*” OR “calculate*” OR “sizing” OR “ measure*”) AND (“Improvement” OR “Progress”) AND (“Process” OR “techniques” OR “models” OR “ approaches”)
3	Agile Mobile Application Development and testing estimation(RQ2.b.)	(“agile” OR “scrum” OR ”XP” OR “lean” Or “crystal”) AND (“Mobile Application” OR “Mobile software” OR “Mobile App” OR “Mobile project”) AND (“Develop*”) AND (“Test*” OR “verification” OR “validation”) AND (“Effort” OR “cost” OR “resource” OR “size” OR “metric” AND (“estimate*” OR “predict*” OR “assessment” OR “forecast*” OR “calculate*” OR “sizing” OR “measure*”)
4	Mobile Application characteristics(RQ3)	(“Mobile Application” OR “Mobile software” OR “ Mobile App” OR “ Mobile project”) AND (“Characteristics” OR “Features” OR “Attribute” OR “Factors”)

2.2.1.2. Data sources. The digital databases that were used to search the keywords are SpringerLink, IEEE Xplore, ACM Digital Library, Elsevier Science Direct, Research Gate, CiteSeer, and InderScience.

2.2.1.3. Search process in data sources. The next phase is to apply the search string to the chosen electronic data sources to find all the entailed studies. This phase is divided into two sub-phases: primary and secondary search phase. In the Primary Search Phase, the electronic data sources identified are examined based on the search string defined earlier. Initially, a total of 2467 results was retrieved with the chosen search string. These results from data sources are monitored to include search string in title and abstracts. The search string is again refined each time to check the outcome and analyzed for better results. Additionally, results are restricted to peer-reviewed conference papers and journal papers. The duplicate titles and abstracts are removed. In the secondary search phase, a technique called snowball tracking is used for studying all the references of primary studies to exploit further studies and increase the chances of inclusion of important papers in the systematic literature review. Table 4 lists the refined results from data sources after primary and secondary search phase.

2.2.2. Inclusion/Exclusion criteria for selecting studies

The results acquired through the various studies generated with the search string defined previously in the electronic databases were analyzed according to the Inclusion/Exclusion criteria. Table 5 enlists the search string category along with inclusion and exclusion criteria.

Both the authors carried the paper selection process independently. The list of studies from primary and secondary search phase is reviewed by authors. The authors then analyze the studies independently and then mark them as In (Include), Un (Uncertain) and Ex (Exclude). The authors followed exclusion criterion through two stages:

Table 4
Overview of search results.

Data Sources	Relevant Search Results
SpringerLink	53
IEEE Xplore	75
ACM Digital Library	57
Elsevier Science Direct	62
ResearchGate	83
Others(Google, ProQuest)	20
InderScience	5
CiteSeer	4
Total	**359**

1. First by reviewing the title and abstract. If title and abstract is in accordance to required information i.e. as per shown in Table5 for each RQ then,
2. Second stage is to review the full text, especially conclusion part.

The list of studies after marking each one with In, Un or Ex from authors is now reviewed collectively. In case of disagreement among authors, the decision on whether to exclude or include the study is based on decision rule table proposed by (Petersen et al., 2015). The decision rule table is exhibited in Table 6. The rules (A to F) against all cases of agreement and disagreement are shown in Table 6.The studies having Ex from both the authors are excluded right away following rule F in the decision table. Rest of the studies following under A to E is included for further analysis. The inclusion and exclusion criteria ended with 78 appropriate papers out of 359.

2.2.3. Quality assessment

To assess the quality of the shortlisted papers; a set of 7 questions is prepared to be answered for each shortlisted paper. The question can be answered as ‘Y = 1′, ‘M = 0.5 ‘or ‘N = 0′. The score of 1 (Y = Yes) means the paper under consideration for quality assessment is explicitly well answered for a particular question (Q1-Q7); score 0.5 (M = Medium) means partially answered and score 0 (N = No) means not answered. The questionnaire was developed by using the guidelines defined by Kitchenham and Charters (2007). Following are the questions in the questionnaire:

Q1. Are the research motives clearly stated?
Q2. Was the study designed to achieve the aims?
Q3. Are the development and testing estimation techniques for mobile apps well defined?
Q4. Are the estimation accuracy measures soundly construed?
Q5. Is the research process documented sufficiently?
Q6. Are all research questions answered sufficiently?
Q7. Are the key findings specified plainly in rapport to credibility, validity, and reliability?

The authors have executed the quality assessment of all the carefully chosen primary studies. Due to the low quality, four papers are excluded from selected studies. Hereafter, 75 papers are designated to report four RQs. The final scores can be seen in Appendix A for 75 selected studies out of 78 along with its study IDs and references. The Study IDs are numbered according to RQ numbers where S stands for Study; RQ stands for Research Question followed by research question number and then identified

Table 5
Inclusion and exclusion criteria.

Search String Category	Included	Excluded
Category 1(RQ1)	Studies related to test estimation techniques for traditional software	Studies related to software development estimation techniques are excluded which do not feature estimation of testing phase for traditional software
Category2 (RQ2.a.)	Studies related to estimation methods for mobile application development and testing	Studies not related to mobile application development and testing estimation are removed
Category3(RQ2.b.)	Studies related to estimation methods in agile mobile application development and testing	Studies not related to agile mobile application development and testing are eliminated
Category 4(RQ3)	Studies that include characteristics of mobile application only	Studies having software characteristic and not mobile software/application characteristic are excluded
Applied to all Categories	Described in English Peer-reviewed papers are selected	Studies not defined in the English Language Not peer reviewed

Table 6
Decision table rules in disagreement case followed from (Petersen et al., 2015).

	Author1			
		Include(In)	Uncertain(Un)	Exclude(Ex)
Author2	Include (In)	A	B	D
	Uncertain(Un)	B	C	E
	Exclude (Ex)	D	E	F

study number. For answering an RQ1 total of 26 studies are devoted from 75 selected studies, 22 studies to RQ2 and finally 27 studies to RQ3.

2.2.4. Data extraction

The data extraction phase elaborates the mining of data from the final selected studies that address the peculiarities of RQs. The data extraction for the finally chosen studies are done in an MS Excel sheet.

2.3. Results reporting

This section defines the results relayed to the systematic literature review questions. The results are presented in a tabular format for each study.

2.3.1. Selected studies overview

Fig. 2 shows the distribution of the chosen studies through the published sources. Out of the 75 studies, 24(32%) came from IEEExplore, 11 studies (14%) came from SpringerLink, ACM Digital Library 14(19%), 11(15%) from Research Gate, 3(4%) studies from CiteSeer, 2(3%) from Elsevier ScienceDirect, 3(4%) from InderScience and 7(9%) from others (ProQuest, GoogleScholar, TU/e Repository, scielo.org.co, SemanticScholar, IGI Global). The distribution of selected studies from different sources is shown in Fig. 3. Maximum papers are referred to the year 2014, 2015, 2016 and one each from 1999, 2001 and 2005. The distribution

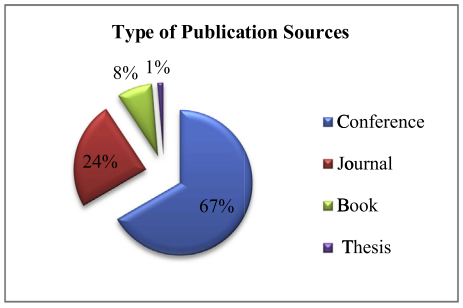

Fig. 2. Publication Sources for Selected Studies.

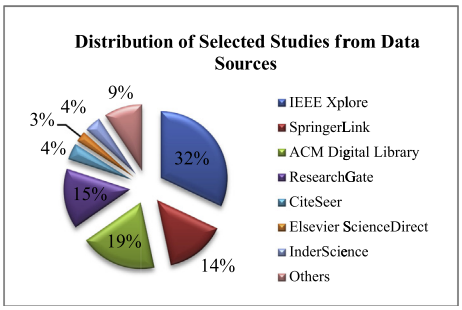

Fig. 3. Distribution of Selected Studies from Data Sources.

of selected studies according to the published year can be seen in Fig. 4.

2.3.2. Results reporting on RQ1

To answer RQ1, out of seventy-five selected studies, twenty-six studies cover all the facets of RQ1.

Test Point Analysis (TPA) proposed by Veenendaal et al. (1999) is based on function point analysis used for estimating the functional size of software with additional attributes such as testing strategy and productivity.

Use Case Point Analysis (UCP) by Nageswaran (2001) examines testing characteristics and their complexity along with software development complexity.

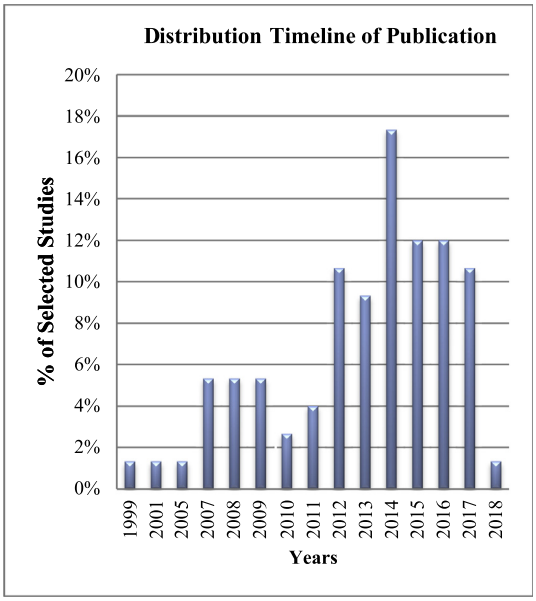

Fig. 4. Distribution of Selected Studies (Year-Wise).

Test Execution Points is a technique proposed by (E Aranha and Borba, 2007a; Eduardo Aranha and Borba, 2007) based on test specification. The test cases are assigned execution points and then the effort is calculated

The model proposed by Abran et al. (2007) for estimating the test volume and effort, the functional requirements are taken as bases to form test estimation and then nonfunctional requirements are taken into consideration.

An approach by Kushwaha and Misra (2008) cognitive information complexity calculation and McCabe's Cyclomatic complexity measure is used to estimate the test execution effort.

Another approach by Zhu et al. (2008a) consist of three attributes for test effort estimation namely test case number, test execution complexity, and tester and then uses a historical database to assign effort.

Another approach by Zhu et al. (2008b) is an extension to existing UCP and considers test execution as a two-dimensional vector having testing experience and knowledge of the application.

The method suggested by Lazić and Mastorakis (2009) covers white box testing and test activities based on Software/System Test Point (STP) metric. The model is implemented on estimating object-oriented software projects by computing size and then applying three steps of COTECOMO model.

A model presented by Silva et al. (2009) is based on historical efficiency data of testers and functional test execution effort estimation and then the model accuracy is measured against different software as a case study.

Another model presented by Abhishek et al. (2010) studies the use of use case and neural network in Precoding phase and then in postcoding phase again neural network with variable, complexity and criticalness component as input is used to calculate test efforts.

In the model presented by Souza and Barbosa (2010) modified TPA is used by making it simpler and hence easy to use. In this model, there are two steps:-one followed by the test designer and second by the tester. Each of the steps has further sub-steps to follow to finally provide test effort estimation.

Aloka et al. (2011) presented an approach which is a combination of UCP, TPA, and particle swarm optimization (PSO) based approach to optimize the test effort estimation.

The approach proposed by Srivastava et al. (2012) is an adaptable model of UCP along with cuckoo search, a *meta*-heuristic approach, for test effort estimation.

A model is proposed by Sharma and Kushwaha (2013) based on SRS, then the complexity of requirements is computed. Requirement based test function points (RBTFP) is calculated based on the complexity of requirements and Technical and Complexity Factors (TEF).

Bhattacharya et al. (2012) proposed an approach which considers features of the software testing effort (STE) estimation by proposing a soft computing technique, Particle Swarm Optimization (PSO) along with COCOMO and test effort drivers into a single stage.

In the approach by Nguyen et al. (2013) the test case point is derived by measuring checkpoints, preconditions, test data and type of testing. The authors compared the proposed approach on two industrial case studies that used the experienced-based approach of testers.

Another heuristic approach by Srivastava et al. (2014) based on bat algorithm used along with existing test effort estimation techniques, UCP and TPA. Later the results obtained after applying bat algorithm are compared with those obtained from UCP and TPA to conclude that findings are improved and nearer to the actual effort using bat algorithm.

In the model proposed by Zapata-Jaramillo and Torres-Ricaurte (2014), the concept of a pre-conceptual schema is used which is a graphical technique to show domain knowledge in a natural language.

An approach by Hauptmann et al. (2014), the test suits are taken as an input and then a cost model is designed based on estimation done by an expert. In the cost model estimation is provided for test suite creation, test suite maintenance, and test execution.

The model presented by Srivastava (2015) uses fuzzy logic and fuzzy multiple linear regression techniques along with COCOMO-II to estimate software test effort. The problem with the model is usability while designing fuzzy rules. However, results produced using this model are better than existing methods.

The method proposed by Arumugam and Babu (2015) is based on the UCM (Use Case Model) and Function Point Analysis (FPA). The use case model is adapted to use case graph and later the edges acting as alternatives for required components are assigned weights. Then FPA is followed for assigning appropriate complexity weights to System Testing Technical Complexity Factors.

An approach by Badri et al. (2015) is used in a model that covers unit testing effort only and forms its bases on Quality Assurance Indicator (Qi) metric along testing effort comprised in scripting unit test cases.

An automatic tool, PredSym, is presented by Bhattacharyya and Malgazhdarov (2016) predicts the code coverage for testing by using a machine learning technique called symbolic execution tool i.e., KLEE.

Islam et al. (2016) demonstrated a web-based test effort estimation tool based on COCOMO-II and have successfully implemented it on 22 projects.

Jin and Jin (2016) proposed an optimization technique called quantum particle swarm optimization (QPSO) algorithm for optimizing parameters in test effort function (TEF) used in Software Reliability Growth Model.

Table 7 lists the summarized review in form of a matrix of all studies selected for studying test effort estimation techniques in traditional software. From the table 7, it can be analyzed that model-based approaches are prominently followed in most of the studies. The tool support for estimation is found hardly in 4 studies. Some studies have only proposed a model and have not validated it on industrial projects.

Table 8 lists the methods used in each study and found that the Use Case Point (UCP) method is prominently used in many studies with extension, modification, and optimization. The second most followed approach is Test Point Analysis (TPA) by several authors. Many studies include a hybrid approach by collaborating with *meta*-heuristic techniques to optimize the estimation process.

They have used different measures to determine the accuracy of the estimation result and most followed accuracy measure is MRE (Magnitude of Relative Error) and MMRE (Mean Magnitude of Relative Error). Table 9 lists the accuracy measures exercised in selected studies.

2.3.3. Results reporting on RQ2

The focus of this research is on mobile applications rather than on traditional applications, RQ2 focuses on elaborating estimation of development and testing of mobile apps in traditional development process and Agile Development process. Out of seventy-five selected studies, twenty-two studies are ardent to answer RQ2.

2.3.3.1. Traditional techniques for estimating mobile application development and testing (RQ2.a.). Seventeen studies out of the selected twenty-two investigated the traditional estimation techniques for mobile applications. There are many development estimation techniques and many testing effort estimation techniques in literature for traditional software. But as the focus is on mobile applications, this study covers development effort and test effort for only mobile software. Table 10 lists the identified techniques

Table 7
Matrix of Study ID and Identified Parameters in Test Estimation for Traditional Software.

Study ID	Model-Based Approach	Hybrid Approach	Metaheuristic Approach	Analogy-Based Approach	Non-Model Based Approach	Experimental study Involved	Accuracy Parameter	Proposed model only	Tool Support
SRQ1-1	X							X	
SRQ1-2	X					X			
SRQ1-3	X							X	
SRQ1-4	X					X	X	X	X
SRQ1-5	X					X	X		
SRQ1-6	X					X			X
SRQ1-7	X						X	X	
SRQ1-8				X	X	X	X		
SRQ1-9	X					X	X		
SRQ1-10	X					X	X		
SRQ1-11	X	X		X		X			
SRQ1-12	X					X	X		
SRQ1-13	X	X	X			X	X		
SRQ1-14	X	X	X			X	X		
SRQ1-15	X					X			
SRQ1-16	X	X	X					X	
SRQ1-17	X					X	X		
SRQ1-18	X	X	X			X	X		
SRQ1-19	X							X	
SRQ1-20					X	X			
SRQ1-21	X	X		X		X	X		
SRQ1-22	X	X				X	X		
SRQ1-23	X					X	X		
SRQ1-24				X		X	X		X
SRQ1-25	X					X	X		X
SRQ1-26			X			X	X		

Table 8
Different test estimation techniques and required input in each technique found in selected studies.

Method	Input	Study ID
Test Point Analysis	Functional Requirements	SRQ1-1
Use Case Point	Functional Requirements	SRQ1-2
Test Execution Point	Test Requirements to form test cases	SRQ1-3,SRQ1-4
Test Volume	Functional and Non Functional Requirements	SRQ1-5
Cognitive information complexity and cyclomatic complexity measure	Line of code	SRQ1-6
Extension To UCP	Functional Requirements and test team knowledge	SRQ1-7
Experience Based	Historical database and tester knowledge	SRQ1-8
Software/system Test Point	LOC, UCP, FP	SRQ1-9
Test execution point and Historical efficiency data	Historical efficiency data	SRQ1-10
Use case and neural network	Requirements	SRQ1-11
Test Point Analysis	Test Designer and tester	SRQ1-12
UCP,TPA,PSO	Software related parameter	SRQ1-13
UCP + cuckoo search	No. of parameters and actual effort	SRQ 1–14
Requirement based test function points (RBTFP) and TEF	SRS	SRQ 1–15
COCOMO + PSO	Functional and non-Functional requirements	SRQ1-16
Test Case Point	Test Cases	SRQ1-17
UCP, TPA, Bat Algorithm	No. of parameters and actual effort	SRQ1-18
Graphical schema	Requirements	SRQ1-19
Cost model	Test Suite	SRQ1-20
COCOMO II, fuzzy logic, and fuzzy multiple linear regression	software requirement specification document	SRQ1-21
System Test Size Points	Use Case Model and Function Point Analysis	SRQ1-22
MLR (Multinomial Logistic Regression) models based on the (Quality Assurance Indicator –Qi metric)	Unit test cases	SRQ1-23
The static program features to predict the coverage explored by KLEE	Code	SRQ1-24
COCOMO –II tool	Functions and use cases	SRQ1-25
Quantum Particle Swarm Optimization (QPSO) Algorithm	Historical data	SRQ1-26

where an agile methodology is not followed for the development of mobile apps. The techniques are broadly classified into three categories by Mendes (2007) i.e. Algorithmic-based models, Expert Judgment based models, and analogy based models. Some of the approaches consider estimation of development and testing of the mobile app as a single process and two studies have considered test estimation of mobile apps as a separate one. COSMIC Function Size Measurement (Abdullah et al., 2014; D'Avanzo et al., 2015; de Souza and Aquino 2014; Ferrucci et al., 2015; Heeringen and Gorp, 2014; Nitze, 2013; Sellami et al., 2016; Vogelezang et al., 2016) is frequently used for estimation technique which is used to measure functional size of the mobile app. Other types of estimation techniques identified are Function Point Analysis (Preuss, 2013; Tunalı, 2014) and Use Case Point (Haoues et al., 2017) which are algorithmic-based models that measure functional, technical factors and environmental factors for estimation. Regression-Based technique (Shahwaiz et al., 2016) uses a parametric model based on effort predictors and data points collected through an online questionnaire which are further used in the regression model. Delphi method (Catolino et al., 2017) is based on experience to

Table 9
Accuracy parameters for test estimation techniques in traditional software.

Accuracy Parameters	Study ID
MRE	SRQ1-4, SRQ1-9,SRQ1-13,SRQ1-14,SRQ1-18,SRQ1-21
MMRE	SRQ1-4, SRQ1-8, SRQ1-9
Mean Absolute Error (MAE)	SRQ1-8, SRQ10
Mean Relative Absolute Error (MRAE)	SRQ1-7, SRQ1-8
Pred(x)	SRQ1-4, SRQ1-7, SRQ1-9
R^2	SRQ1-5, SRQ1-26
Compared with Actual Effort	SRQ-12,SRQ1-17,SRQ1-23,SRQ1-25
Others	SRQ1-15(comparison with others UCP, Test specification, Scenario-based, Test execution effort, Experience-Based approach, CICM), SRQ1-22(*t*-test), SRQ1-23 (Comparison with LOC)

Table 10
Traditional estimation techniques for mobile applications.

Study ID	Traditional Estimation Techniques	Approach Type	Estimation Covers	
			Development and Testing Together	Testing as a separate process
SRQ2-1	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-2	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-3	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-4	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-5	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-6	Function Point Analysis	Algorithmic-based Model	✓	
SRQ2-7	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-8	COCOMO –I and II	Algorithmic-based Model	✓	
SRQ2-9	Delphi method	Expert Judgment	✓	
SRQ2-10	Use Case Point	Algorithmic-based Model	✓	
SRQ2-11	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-12	Regression-Based	Algorithmic-based Model	✓	
SRQ2-13	Hybrid (Analogy based estimation + Function Size Measurement)	Analogy and Algorithmic based model	✓	
SRQ2-14	Architecture-Based	Algorithmic-based Model		✓
SRQ2-15	Function Point Analysis	Algorithmic-based Model	✓	
SRQ2-21	COSMIC Function Size Measurement	Algorithmic-based Model	✓	
SRQ2-22	Test sizeand Execution Complexity Measure	Algorithmic-based Model		✓

estimate the effort whereas Architecture Based estimation model (Wadhwani et al., 2008) for reliability and testing estimation of the mobile application is proposed and the case study was conducted in two companies. Another algorithmic approach for estimating the cost of developing Android mobile apps are based on COCOMO –I and II model (Asghar et al., 2016). Analogy-based estimation plus functional size measurement (Nitze et al., 2014) approach is also proposed for mobile apps. One approach (E Aranha and Borba, 2007b) covers estimation of test execution effort taking a risk and test factors as a major contributing feature in estimation and taken mobile application as a case study.

2.3.3.2. Agile techniques for estimating mobile application development and testing (RQ2.b.). The agile methodology aims at facilitating software development processes where changes are acceptable at any stage and provide a structure for highly collaborative software development. In such a dynamic environment, estimation is very challenging (Usman et al., 2014).

Agile approach to mobile application development estimation has very less number of studies. One of the reason could be the adoption of agile to mobile context is still in its evolving phase. The identified studies are listed in Table 11. It can be seen that only one study has proposed a technique for test effort estimation for the mobile app in an agile environment. Other studies consider development and testing estimation together as a single practice.

Traditional use case point method of estimation is extended by adding efficiency and risk factor of testers in the agile team (Parvez, 2013). Another technique (Francese et al., 2015) is based on a stepwise linear regression model which estimate the effort for Android apps from requirements specification including a number of use cases, actors, etc. User story point (Aslam et al., 2017) is refined by considering additional factors along with size and complexity. The quality factor, Novelty factor and Type factor of User Story are added to deliver the best estimation of mobile application development. Additional approach (Qi and Boehm, 2017) uses Early Use Case Point (EUCP) and Extended Use Case Point (EXUCP) along with COCOMO drivers at different iteration levels in agile mobile app development. An experience-driven approach using Delphi technique (Lusky et al., 2018) is used for effort estimation in which mobile app is taken as case studies.

Table 11
Agile estimation techniques for mobile apps.

Study ID	Agile Estimation Techniques	Approach Type	Estimation Covers	
			Development and Testing Together	Testing as a separate process
SRQ2-16	Use Case Point	Algorithmic-based models		✓
SRQ2-17	Step-wise Linear Regression	Algorithmic-based models	✓	
SRQ2-18	User story Point	Expert Judgment	✓	
SRQ2-19	Use Case Point + COCOMO	Algorithmic-based models	✓	
SRQ2-20	Delphi	Expert Judgment	✓	

The estimation attributes identified in the selected studies are mostly focused on size metric whether based on use case, function point and story point. Table 12 lists the other estimation attributes that are known for estimation.

Table 13 lists the parameters used to assess the accuracy of estimation of mobile applications. MMRE and Pred(x) are highly followed in most of the studies dealing with mobile applications.

2.3.4. Results reporting on RQ3

The results from Systematic Literature Review (SLR) recognized 15 characteristics in the majority of the chosen studies after passing all the selection criteria from primary studies. Twenty-seven studies are dedicated to answering RQ3 out of the total seventy-five selected studies. Some of the characteristics are deliberated as characteristics of the mobile device (Limited RAM, Battery, Memory, and Screen size) however many studies emphasize that they need to be considered as these limitations are directed linked to mobile apps development and testing. Fig. 5 shows the type of mobile app characteristics mentioned in selected studies. Table 14 lists the studies in which each characteristic is discussed. The findings of this SLR for RQ3 clearly state that how these mobile app characteristics are different from traditional software. So the development and testing estimation techniques reported in RQ1 and RQ2 does not consider these important characteristics undertaking the estimation process.

2.3.4.1. Description of mobile app characteristics.

- Limited Memory: - The internal memory of the mobile device is limited. The mobile app consumes a memory space when it is installed on the device. The developers should use such programming practices that allow development of small size apps. The testers should check how the app performs when the memory of the device reaches maximum memory limit (Amalfitano et al., 2011; Cao et al., 2012; Charland and Leroux, 2011; Dantas et al., 2009; Kim et al., 2009; Kim, 2012; Liu et al., 2014; Lu et al., 2012; Muccini et al., 2012; Vilkomir and Amstutz, 2014; Zein et al., 2016, 2015).
- Limited CPU or Small Processing capacity: - As the mobile devices have small processors, the mobile apps should be developed and tested in a way so as to decipher the consumption of the processor while it runs on the mobile device (Amalfitano et al., 2011; Cao et al., 2012; Charland and Leroux, 2011; Ciman and Gaggi, 2017; Dantas et al., 2009; Kim, 2012; Liu et al., 2014; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2016; Zhang and Adipat, 2005).

Table 12
Estimation attributes for mobile applications.

Estimation Attributes	Study ID
Size	SRQ2-1, SRQ2-2,SRQ2-3,SRQ2-4,SRQ2-5,SRQ2-6,SRQ2-7, SRQ2-10,SRQ2-11,SRQ2-13,SRQ2-15,SRQ2-16,SRQ2-17, SRQ2-18(user stories),SRQ2-19,SRQ2-21,SRQ2-22
Cost	SRQ2-8,SRQ2-13,SRQ2-19
Others	SRQ2-9(Score Metric), SRQ2-12(Mean and SD of collected mobile apps variables), SRQ2-14(architecture based), SRQ2-20

Table 13
Parameters for measuring estimation accuracy.

Accuracy parameters	Study ID
MRE(Magnitude of Relative Error)	SRQ2-2,SRQ2-3,SRQ2-18,SRQ2-22
MMRE(Mean Magnitude of Relative Error)	SRQ2-2,SRQ2-3,SRQ2-12,SRQ2-18,SRQ2-19,SRQ2-22
MdMRE(Median MRE)	SRQ2-2,SRQ2-3, SRQ2-12
Pred(percentage relative error deviation)	SRQ2-2,SRQ2-3, SRQ2-12,SRQ2-18,SRQ2-19,SRQ2-22
Linear Regression (R^2)	SRQ2-12,SRQ2-19
Not Defined	SRQ2-1,SRQ2-4,SRQ2-6,SRQ2-7,SRQ2-9SRQ2-10,SRQ2-13,SRQ2-14,SRQ2-20,SRQ2-21
Others	SRQ2-8(web-based survey), SRQ2-11(Compared with actual effort), SRQ2-15(Compared with actual effort), SRQ2-16(Comparison with actual effort), SRQ2-17(compared with source code as a software measure)

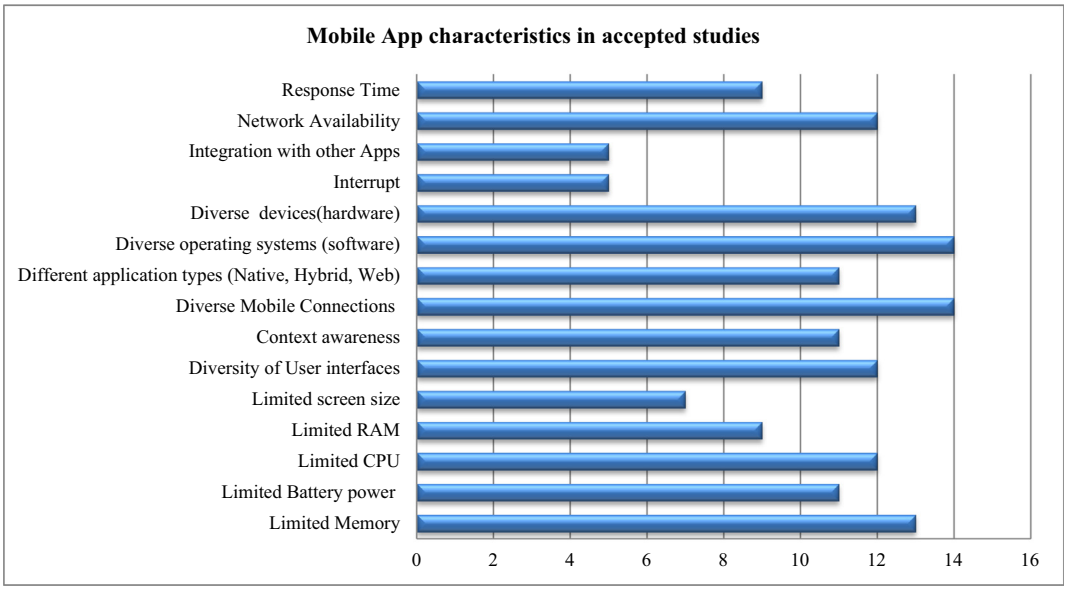

Fig. 5. Mobile Application Characteristics in accepted papers.

Table 14
Identified mobile Application characteristics in various studies.

Mobile App characteristic	Study ID
Limited Memory	SRQ3-1, SRQ3-3, SRQ3-4, SRQ3-5, SRQ3-6, SRQ3-7, SRQ3-9,SRQ3-10, SRQ3-14, SRQ3-16, SRQ3-19, SRQ3-22, SRQ3-25
Limited Battery power	SRQ3-1, SRQ3-2, SRQ3-4, SRQ3-6, SRQ3-7, , SRQ3-9, SRQ3-10, SRQ3-14, SRQ3-22, SRQ3-25, SRQ3-26
Limited CPU	SRQ3-1, SRQ3-2, SRQ3-4, SRQ3-5, SRQ3-6, SRQ3-7, SRQ3-9, SRQ3-10, SRQ3-14, SRQ3-22, SRQ3-25, SRQ3-26
Limited RAM	SRQ3-4, SRQ3-6, SRQ3-7, SRQ3-9, SRQ3-10, SRQ3-14, SRQ3-22, SRQ3-25, SRQ3-26
Limited screen size	SRQ3-2, SRQ3-15, SRQ3-16, SRQ3-17, SRQ3-20, SRQ3-24, SRQ3-26
Diversity of User interfaces(touchscreen, keypad, voice)	SRQ3-2, SRQ3-5, SRQ3-6, SRQ3-9, SRQ3-12,SRQ3-14, SRQ3-15, SRQ3-17, SRQ3-19, SRQ3-22, SRQ3-27
Context awareness	SRQ3-2, SRQ3-4, SRQ3-5, SRQ3-6, SRQ3-12, SRQ3-15, SRQ3-22, SRQ3-24,SRQ3-25,SRQ3-26, SRQ3-27
Diverse Mobile Connections (2G, 3G, 4G and various wireless networks)	SRQ3-1, SRQ3-2, SRQ3-3, SRQ3-4, SRQ3-6, SRQ3-8, SRQ3-10, SRQ3-11, SRQ3-12, SRQ3-13, SRQ3-15, SRQ3-18, SRQ3-19, SRQ3-26
Different application types (Native, Hybrid, Web)	SRQ3-1, SRQ3-3, SRQ3-4, SRQ3-6, SRQ3-7, SRQ3-10, SRQ3-11, SRQ3-14, SRQ3-16, SRQ3-19, SRQ3-26
Diverse operating systems (software)	SRQ3-6, SRQ3-9, SRQ3-10, SRQ3-11, SRQ3-13, SRQ3-15, SRQ3-16, SRQ3-19, SRQ3-20, , SRQ3-21, SRQ3-22, SRQ3-23, SRQ3-25, SRQ3-26
Diverse devices(hardware)	SRQ3-3, SRQ3-6, SRQ3-8, SRQ3-9, SRQ3-10, SRQ3-13, SRQ3-16, SRQ3-20, SRQ3-21, SRQ3-22, SRQ3-23, SRQ3-25, SRQ3-26
Interrupt	SRQ3-4, SRQ3-13, SRQ3-23, SRQ3-26, SRQ3-27
Integration with other Apps	SRQ3-1, SRQ3-4, SRQ3-6, SRQ3-11, SRQ3-23
Network Availability	SRQ3-1, SRQ3-2, SRQ3-3, SRQ3-6, SRQ3-8, SRQ3-10, SRQ3-12, SRQ3-15, SRQ3-18, SRQ3-19, SRQ3-24, SRQ3-26
Response Time	SRQ3-1, SRQ3-3, SRQ3-6, SRQ3-7, SRQ3-10, SRQ3-14, SRQ3-16, SRQ3-19, SRQ3-26

- Limited RAM: - Apps should be programmed and tested so that they exhaust less amount of memory when they run on the mobile device. Large size mobile apps tend to run slow and further influence user experience (Cao et al., 2012; Charland and Leroux, 2011; Ciman and Gaggi, 2017; Kim, 2012; Liu et al., 2014; Lu et al., 2012; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2016).
- Limited screen size and Orientation: - Mobile devices have a small screen. Keeping the constraint in mind the app should be developed and tested well to check if it operates differently on varied screen size and orientation (Arzenšek and Heričko, 2014; Costa et al., 2014; Holl and Elberzhager, 2016; M. Amen et al., 2015; Nidagundi and Novickis, 2017; Vilkomir and Amstutz, 2014; Zhang and Adipat, 2005).
- Limited Battery: – Mobile devices have very limited battery life. The mobile apps should be developed in a way so they should consume less battery power. The app should be tested in a scenario when the battery is too low that how it behaves in this instance and should retain data integrity when the battery dies (Amalfitano et al., 2011; Cao et al., 2012; Dantas et al., 2009; Kim, 2012; Liu et al., 2014; Lu et al., 2012; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2016; Zhang and Adipat, 2005).
- The diversity of User interfaces (touchscreen, keypad, and voice):- As input to a mobile device can be through voice, touch keypad, stylus, etc., the mobile app should be tested against all input interfaces (Arzenšek and Heričko, 2014; Charland and Leroux, 2011; Costa et al., 2014; de Cleva Farto and Endo, 2017; Kim, 2012; Kirubakaran and Karthikeyani, 2013; Liu et al., 2014; Muccini et al., 2012; Zein et al., 2016, 2015; Zhang and Adipat, 2005).
- Context awareness: - Mobile apps can react variedly based on their environment which means that the app should be tested to take into account all the input explicitly delivered by operators and likewise implicit input regarding physical and computational context of operators (Arzenšek and Heričko, 2014; Charland and Leroux, 2011; Ciman and Gaggi, 2017; Holl and Elberzhager, 2016; Kirubakaran and Karthikeyani, 2013; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2016; Zhang et al., 2015).
- Diverse Mobile Connections (2G, 3G, 4G and various wireless networks), Mobile network operators and user's mobility: – Mobile app should be tested under all different connections such as Wireless networks, Bluetooth, 3G, 4G, NFC, etc., (Arzenšek and Heričko, 2014; Charland and Leroux, 2011; Dantas et al., 2009; Franke et al., 2012; Giessmann et al., 2012; Göth, 2015; Kim et al., 2009; Kirubakaran and Karthikeyani, 2013; Lu et al., 2012; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2015; Zhang and Adipat, 2005).
- Different application types (Native, Hybrid, and Web):- The development and testing of native, web and hybrid mobile application is different. So each one should be tested thoroughly depending on the type of app (Cao et al., 2012; Charland and Leroux, 2011; Dantas et al., 2009; Giessmann et al., 2012; Kim et al., 2009; Liu et al., 2014; Lu et al., 2012; Muccini et al., 2012; Nidagundi and Novickis, 2017; Vilkomir and Amstutz, 2014; Zein et al., 2015).
- Diverse operating systems (software):- The mobile apps run on the particular operating system. There are various mobile OS such as iOS, Android, RIM, Windows, and Symbian etc. The app should be tested for the required platforms for proper compatibility (Ciman and Gaggi, 2017; Giessmann et al., 2012; Kim, 2012; Lu et al., 2012; Amen et al., 2015; Muccini et al., 2012; Nidagundi and Novickis, 2017; Umuhoza and Brambilla, 2016; Vilkomir and Amstutz, 2014; Zein et al., 2015, 2016; Zhang et al., 2015).
- Diverse devices (hardware):- Mobile devices get launched in the market every now and then with a change in technology. The mobile App should be tested for maximum no. of devices wherever possible (Ciman and Gaggi, 2017; Franke et al., 2012; Kim et al., 2009; Kim, 2012; Kirubakaran and Karthikeyani, 2013; Lu et al., 2012; Amen et al., 2015; Nidagundi and Novickis, 2017; Vilkomir and Amstutz, 2014; Zein et al., 2016; Zhang et al., 2015).
- Interrupt: -The mobile app should be tested for all kind of interruptions such as receiving a message, battery low, in between calls; while it is running on the mobile device (Charland and Leroux, 2011; Dalmasso et al., 2013; de Cleva Farto and Endo, 2017; Nidagundi and Novickis, 2017; Umuhoza and Brambilla, 2016).
- Integration with other Apps: - There are some apps that run in integration with other apps. Testing should be done to check if mobile app integrates well with other apps on the user's device or not (Charland and Leroux, 2011; Dantas et al., 2009; Giessmann et al., 2012; Muccini et al., 2012; Umuhoza and Brambilla, 2016).

- Network Availability: - Network availability varies, so apps should be developed and tested keeping this constraint in mind. It should be tested how it behaves when the user moves to the remote area when networks are not in range (Arzenšek and Heričko, 2014; Dantas et al., 2009; Franke et al., 2012; Göth, 2015; Holl and Elberzhager, 2016; Kim et al., 2009; Kirubakaran and Karthikeyani, 2013; Muccini et al., 2012; Nidagundi and Novickis, 2017; Zein et al., 2015; Zhang et al., 2015).
- Response Time: - The mobile app should be tested for its start time which should be immediate through any input interface means (Cao et al., 2012; Dantas et al., 2009; Kim et al., 2009; Liu et al., 2014; Nidagundi and Novickis, 2017; Vilkomir and Amstutz, 2014; Zein et al., 2015).

3. Discussion, research Gap, and future work

The results from SLR for answering RQ1 indicate that the model-based approach is followed by many researchers. The base input to most of the traditional test estimation techniques is functional requirements that are then used to derive functional size. Use Case Point; its extension and optimizations are prominently exploited among all the identified techniques in traditional software test estimation. The tool support for test effort estimation is very limited. In order to measure the accuracy of the estimation techniques, the estimated effort is compared to the actual effort. Apart from this, the other statistical measure MRE and MMRE are also widely accepted.

Answers for RQ2 regarding estimation techniques for development and testing of mobile apps are identified both in traditional software development environment and agile software development. In mobile applications testing estimation algorithmic-based models are prevalent. COSMIC FSM techniques are preferred due to its designing in such a way that it could be applied in a very broad range of architectures, including mobile apps. COSMIC FSM provides the functional size of the software which is used to derive effort for estimation of mobile app development. Majority of reported studies are contributed towards estimation on the development and for testing effort estimation, only two studies are reported. According to Jayakumar and Abran, (2013), COSMIC size-based estimation models can be used to estimate efforts for the full life cycle of software development or any major life cycle activities such as Testing. This can serve as a future direction when instigated in the mobile domain for proposing a standardized model and validate the estimation results of the model on mobile apps.

Adoption of agile to mobile context is still in its evolving phase. From Section 2.3.3.2 it can be perceived that very less number of studies is proposed in test effort estimation for the mobile app in an agile environment. Algorithmic-based models are reported mainly in three identified studies. The main attribute in the estimation of mobile app reported in maximum identified studies is size. The size is measured in terms of cosmic function point, Function point, use case point, and user story point. For measuring the accuracy of estimation models in mobile app domain, MMRE and Pred(x) are reported by most studies.

After identifying test estimation techniques for traditional software and mobile software, a comparative chart is shown in Table 15. The comparison is formed on the basis of findings in SLR.

In answer to RQ3, fifteen characteristics are identified in section 2.3.4. Testing of the mobile app on "different mobile OS" is identified in maximum studies along with testing on "different mobile connections". "Limited memory" of mobile devices which is rather a mobile device constraint is also identified as mobile app characteristic in many studies as how much memory a mobile app consume while running on the device poses a testing constraint too. Other characteristics are discussed in Section 2.3.4.1. As for estimation of testing of the mobile app, these identified characteristics may or may not affect the test estimation process. The impact of each characteristic while performing test estimation can range from being negligible to highly significant. A survey on investigating the impact of mobile app characteristics, accumulated from mobile app developers and testers can be beneficial to accomplish this task. This identified research gap can be considered as probable research direction for future work.

4. Threats to validity

The validity threats for SLR are discussed in this section. The *construct validity threat* in terms of SLR (Systematic Literature Review) is its failure to claim coverage of all relevant studies. By adopting a good search strategy and using all relevant search strings and their synonyms we tried to mitigate this threat. Also, only one study each from year 1999 and 2001 is selected as they

Table 15
Comparison of test estimation techniques for traditional software and for mobile software/application.

	Prominent Type of Approach Followed	Inputs to Different Techniques	Effort Drivers	Accuracy Parameters	Tool Support
Traditional Software Test Estimation Techniques	Model-Based Approach, Hybrid Approach, Metaheuristic Approach, Analogy-Based Approach, Non-Model Based Approach	SRS(Software Requirement Specification) document, Test Requirements, Historical database, Test Cases	Size, the complexity of environmental factors and technical factors	MRE, MMRE, Mean Absolute Error (MAE), Mean Relative Absolute Error (MRAE), Pred(x), $R^{2,}$ Compared with Actual Effort	Available for few techniques
Mobile Application Development and Test Estimation Techniques	Algorithmic-based models, Expert Judgment, Analogy and Algorithmic based model	Functional user requirements, UML model for collecting functional requirements, SRS, online quote forms from companies, Base Functional Components, functional and quality attributes, historical projects, software artifacts (functional components), test specification written in natural language	Size, risk factor, efficiency factor, Number of screens, Application complexity, Number of 3rd party Apps, Average Flexibility of 3rd party Apps, Server Configuration Flexibility, Memory Optimization Complexity, Connectivity Handling, Test execution complexity	MRE(Magnitude of Relative Error), MMRE(Mean Magnitude of Relative Error), MdMRE(Median MRE),Pred (percentage relative error deviation), R^2 , Compared with Actual Effort	Available for development effort estimation only but none for test effort estimation

represented major techniques for test effort estimation in traditional software which are further enhanced and modified by other authors. Rest of the selected studies for reporting test effort estimation in traditional software is based from year 2007 till 2016 i.e. last decade.

Internal Validity threat concerns with data extraction phase of SLR. The data is extracted from the selected studies by both the authors discretely in excel sheets according to the structure depicted in Appendix C. Later, the extorted data was assessed and some disagreements were discussed among authors. But still, the correctness of extraction by authors poses the internal validity threat.

External validity threat deals with incapability of deriving to the generalized conclusions of SLR results. The characteristics reported in section 2.3.4 tried to conclude maximum features from selected studies but there may be others which can be further investigated as the list is not exhaustive. The authors tried to summarize the findings of SLR from different aspects of estimation techniques but still, it might miss the in-depth analysis of the results.

5. Conclusion

This study investigates the current state-of-art on test effort estimation in traditional software and mobile software/ application in traditional software development process and agile software development by means of Systematic Literature Review (SLR). During SLR, 75 studies are selected, searched from nine online data sources to answer four Research Questions (RQs). The main findings of the survey for RQ1 resulted in providing 26 test estimation techniques for traditional software centered on model-based approach, hybrid approach, *meta*-heuristic approach, analogy-based approach, and non-model based approach. Use Case Point and its extension and optimizations are prominently used in traditional software test estimation. But for the mobile application domain, a COSMIC method for function size estimation is prevailing in the literature survey. For estimation techniques reported in section 2.3.3.2 for agile mobile application development and testing, sturdy conclusions cannot be drawn due to lack of endemic studies in the literature. But results from section 2.3.3.1 on COSMIC FSM method for mobile applications can be further explored in an agile environment based on discussions presented by the Kamal Ramasubramani (2016) that COSMIC FSM can be investigated for estimating testing efforts in agile projects. A comparison of test estimation techniques for traditional software and mobile application software is presented based on SLR. Later, mobile application characteristics are identified in SLR. It was comprehended that the mobile application characteristics are not enclosed by the present estimation techniques. There is no formal model that exclusively considers mobile application development and testing different from other traditional applications. Lastly, discussions and some research gaps are reported and certain future research avenues in the estimation of mobile apps in agile and traditional development and testing process are discussed.

Acknowledgements

Authors are highly thankful to IK Gujral Punjab Technical University, Kapurthala, Punjab, India for providing the opportunity to conduct this research work.

Conflicts of interest

The authors declare no conflicts of interest.

Appendix A

Study ID	References	Scores							Study ID	References	Scores							Study ID	References	Scores						
		Q1	Q2	Q3	Q4	Q5	Q6	Q7			Q1	Q2	Q3	Q4	Q5	Q6	Q7			Q1	Q2	Q3	Q4	Q5	Q6	Q7
SRQ1-1	(Veenendaal and Dekkers, 1999)	1	1	1	0.5	1	1	0	SRQ2-1	(Nitze, 2013)	1	1	0.5	1	0.5	0.5	0.5	SRQ3-1	(Dantas et al., 2009)	0.5	0.5	1	1	0.5	1	1
SRQ1-2	(Nageswaran, 2001)	1	0.5	0.5	0.5	1	0	1	SRQ2-2	(D'Avanzo et al., 2015)	1	0	1	0.5	0.5	1	1	SRQ3-2	(Zhang and Adipat, 2005)	1	1	1	0.5	0.5	1	1
SRQ1-3	(E Aranha and Borba, 2007a)	1	1	0.5	1	0.5	0.5	0.5	SRQ2-3	(Ferrucci et al., 2015)	0.5	0.5	1	1	0.5	1	1	SRQ3-3	(Kim et al., 2009)	1	1	0.5	1	0.5	0.5	0.5
SRQ1-4	(Eduardo Aranha and Borba, 2007)	0.5	0.5	1	1	0.5	1	1	SRQ2-4	(Abdullah et al., 2014)	1	1	1	0.5	1	0.5	1	SRQ3-4	(Charland and Leroux, 2011)	0.5	1	0.5	1	0.5	1	0.5
SRQ1-5	(Abran et al., 2007)	1	1	1	0	0.5	1	1	SRQ2-5	(Sellami et al., 2016)	0.5	0	1	0	1	0.5	1	SRQ3-5	(Amalfitano et al., 2011)	1	1	1	0.5	1	0.5	1
SRQ1-6	(Kushwaha and Misra, 2008)	0.5	1	0.5	1	0.5	1	0.5	SRQ2-6	(Preuss, 2013)	1	0.5	0.5	0.5	1	0	1	SRQ3-6	(Muccini et al., 2012)	1	1	1	0.5	1	1	0
SRQ1-7	(Zhu et al., 2008b)	1	1	1	0.5	1	0.5	1	SRQ2-7	(Heeringen and Gorp, 2014)	1	0.5	1	1	1	0.5	0.5	SRQ3-7	(Cao et al., 2012)	0.5	1	1	1	1	0.5	1
SRQ1-8	(Zhu et al., 2008a)	0.5	1	1	1	1	0.5	1	SRQ2-8	(Asghar et al., 2016)	1	1	1	0.5	0.5	1	1	SRQ3-8	(Franke et al., 2012)	1	0.5	1	1	1	0.5	0.5
SRQ1-9	(Lazić and Mastorakis, 2009)	1	0.5	1	1	1	0.5	0.5	SRQ2-9	(Catolino et al., 2017)	1	1	0.5	1	0.5	0.5	0.5	SRQ3-9	(Kim, 2012)	0.5	0	1	1	0.5	1	1
SRQ1-10	(Silva et al., 2009)	1	1	0.5	1	0	0.5	0.5	SRQ2-10	(Haoues et al., 2017)	1	1	0.5	1	0.5	0.5	0.5	SRQ3-10	(Lu et al., 2012)	1	1	1	0.5	0.5	1	1
SRQ1-11	(Abhishek et al., 2010)	1	1	1	0.5	0.5	1	1	SRQ2-11	(de Souza and Aquino 2014)	1	1	1	0.5	1	1	0	SRQ3-11	(Giessmann et al., 2012)	0.5	1	0.5	1	0.5	1	0.5
SRQ1-12	(Souza and Barbosa, 2010)	1	0	0.5	1	0.5	0.5	0.5	SRQ2-12	(Shahwaiz et al., 2016)	0.5	1	1	0	1	0.5	1	SRQ3-12	(Kirubakaran and Karthikeyani, 2013)	0.5	1	1	1	1	0.5	1
SRQ1-13	(Aloka et al., 2011)	1	0.5	0.5	0.5	1	0	1	SRQ2-13	(Nitze et al., 2014)	1	1	0.5	1	0.5	0.5	0.5	SRQ3-13	(Dalmasso et al., 2013)	1	1	1	0.5	0.5	1	0
SRQ1-14	(Srivastava et al., 2012)	0.5	1	1	1	1	0.5	1	SRQ2-14	(Wadhwani et al., 2008)	1	1	1	0.5	0.5	1	1	SRQ3-14	(Liu et al., 2014)	1	1	0.5	1	0.5	0.5	0.5
SRQ1-15	(Sharma and Kushwaha, 2013)	0.5	0.5	1	1	0.5	1	1	SRQ2-15	(Tunalı, 2014)	1	0.5	1	1	1	0.5	0.5	SRQ3-15	(Arzenšek and Heričko, 2014)	1	1	0	1	0.5	0.5	0.5
SRQ1-16	(Bhattacharya et al., 2012)	0.5	1	0.5	1	0.5	1	0.5	SRQ2-16	(Parvez, 2013)	1	1	1	0.5	1	0.5	1	SRQ3-16	(Vilkomir and Amstutz, 2014)	1	0.5	0.5	0.5	1	0	1
SRQ1-17	(Nguyen et al., 2013)	1	1	1	0.5	1	0.5	1	SRQ2-17	(Francese et al., 2015)	1	0.5	0.5	0.5	1	0	1	SRQ3-17	(Costa et al., 2014)	1	0.5	1	1	1	0.5	0.5

(continued)

Study ID	References	Scores							Study ID	References	Scores						
		Q1	Q2	Q3	Q4	Q5	Q6	Q7			Q1	Q2	Q3	Q4	Q5	Q6	Q7
SRQ1-18	(Srivastava et al., 2014)	0.5	1	1	0	1	0.5	1	SRQ2-18	(Aslam et al., 2017)	0.5	1	0.5	1	0.5	1	0.5
SRQ1-19	(Zapata-Jaramillo and Torres-Ricaurte, 2014)	1	1	1	0.5	1	1	0	SRQ2-19	(Qi and Boehm, 2017)	0.5	1	1	0	1	0.5	1
SRQ1-20	(Hauptmann et al., 2014)	1	1	1	0.5	0.5	1	1	SRQ2-20	(Lusky et al., 2018)	0.5	0.5	1	1	0.5	1	1
SRQ1-21	(Srivastava, 2015)	1	1	1	0.5	1	0.5	1	SRQ2-21	(Vogelezang et al., 2016)	0.5	1	0.5	1	0.5	1	0.5
SRQ1-22	(Arumugam and Babu, 2015)	1	0.5	0.5	0.5	1	0	1	SRQ2-22	(E Aranha and Borba, 2007b)	1	1	1	0.5	0.5	1	1
SRQ1-23	(Badri et al., 2015)	0.5	1	0.5	1	0.5	1	0.5									
SRQ1-24	(Bhattacharyya and Malgazhdarov, 2016)	1	1	0.5	1	0	0.5	0.5									
SRQ1-25	(Islam et al., 2016)	0.5	0.5	1	1	0.5	1	1									
SRQ1-26	(Jin and Jin, 2016)	1	1	1	0.5	0.5	1	0									

Study ID	References	Scores							Study ID	References	Scores						
		Q1	Q2	Q3	Q4	Q5	Q6	Q7			Q1	Q2	Q3	Q4	Q5	Q6	Q7
SRQ3-18	(Göth, 2015)	0.5	1	1	1	1	0.5	1	SRQ3-20	(M. Amen et al., 2015)	0.5	0.5	1	1	0.5	1	1
SRQ3-19	(Zein et al., 2015)	1	1	1	0.5	1	0	0	SRQ3-21	(Zhang et al., 2015)	1	1	1	0.5	1	0.5	1
SRQ3-22	(Zein et al., 2016)	1	1	0.5	0	0.5	0.5	0.5	SRQ3-23	Umuhoza and Brambilla, 2016	1	1	1	0.5	0.5	1	1
SRQ3-24	Holl and Elberzhager, 2016	0.5	1	0.5	1	0.5	1	0.5	SRQ3-25	Ciman and Gaggi, 2017	0.5	0.5	1	1	0.5	1	1
SRQ3-26	Nidagundi and Novickis, 2017	1	0.5	0	0.5	1	0	1	SRQ3-27	de Cleva Farto and Endo, 2017	0.5	0.5	0.5	0	0.5	0	0.5

References

Abdullah, N.A.S., Rusli, N.I.A., Ibrahim, M.F., 2014. Mobile game size estimation: COSMIC FSM rules, UML mapping model and Unity3D game engine, in: ICOS 2014–2014 IEEE Conference on Open Systems. pp. 42–47.

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.

Abhishek, C., Kumar, V.P., Vitta, H., Srivastava, P.R., 2010. Test effort estimation using neural network. J. Softw. Eng. Appl. 03, 331–340.

Abran, A., Garbajosa, J., Cheikhi, L., 2007. Estimating the test volume and effort for testing and verification & validation. IWSM-MENSURA Conference, 216–234.

Aloka, S., Singh, P., Rakshit, G., Srivastava, P.R., 2011. Test effort estimation-particle swarm optimization based approach. Commun. Comput. Inform. Sci., 463–474

Amalfitano, D., Fasolino, A.R., Tramontana, P., 2011. A GUI crawling-based technique for android mobile application testing, in: Proceedings – 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2011. pp. 252–261.

Aranha, E., Borba, P., 2007a. Test Effort Estimation Models Based on Test Specifications, in: Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, 2007. TAICPART-MUTATION 2007. pp. 1–5.

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, 107–116.

Aranha, E., Borba, P., 2007b. Empirical studies of test execution effort estimation based on test characteristics and risk factors, in: Doctoral Symposium on Empirical Software Engineering (IDoESE 2007).

Arumugam, C., Babu, C., 2015. Test size estimation for object oriented software based on analysis model. J. Softw. 10, 713–729.

Arzenšek, B., Heričko, M., 2014. Criteria for selecting mobile application testing tools. CEUR Workshop Proceed., 1–8

Asghar, M.Z., Habib, A., Habib, A., Zahra, S.R., Ismail, S., 2016. AndorEstimator: android based software cost estimation application. arXiv Prepr arXiv 14, 192–202.

Aslam, W., Ijaz, F., Lali, Muhammad Ikram, Mehmood, Waqar, 2017. Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development. J. Inf. Sci. Eng. 33 (6), 1481–1500.

Badri, M., Toure, F., Lamontagne, L., 2015. Predicting unit testing effort levels of classes: an exploratory study based on multinomial logistic regression modeling. Proced. Comput. Sci., 529–538

Bhattacharya, P., Srivastava, P.R., Prasad, B., 2012. Software test effort estimation using particle swarm optimization. Adv. Intell Soft Comput. 132 AISC, 827–835.

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.

Cao, G., Yang, J., Zhou, Q., Chen, W., 2012. Software Testing Strategy for Mobile Phone, Advances and Applications in Mobile Computing. InTech.

Catolino, G., Salza, P., Gravino, C., Ferrucci, F., 2017. A Set of Metrics for the Effort Estimation of Mobile Apps, in: Proceedings - 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2017. pp. 194–198.

Charland, A., Leroux, B., 2011. mobile application Development : Web vs . native. Commun. ACM.

Ciman, M., Gaggi, O., 2017. An empirical analysis of energy consumption of cross-platform frameworks for mobile development. Pervasive Mob. Comput. 39, 214–230.

Costa, P., Paiva, A.C.R., Nabuco, M., 2014. Pattern based GUI testing for mobile applications, in: Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology. QUATIC, 66–74.

Cunha, T.F.V.D., Dantas, V.L.L., Andrade, R.M.C., 2011. SLeSS: A scrum and lean six sigma integration approach for the development of sofware customization for mobile phones, in: Proceedings - 25th Brazilian Symposium on Software Engineering. SBES, 283–292.

D'Avanzo, L., Ferrucci, F., Gravino, C., Salza, P., 2015. COSMIC functional measurement of mobile applications and code size estimation, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing. - SAC '15, 1631–1636.

Dalmasso, I., Datta, S.K., Bonnet, C., Nikaein, N., 2013. Survey, comparison and evaluation of cross platform mobile application development tools, in: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 323–328.

Dantas, V.L.F.G.M., da Costa, A.L., Andrade, R.M.C., 2009. Testing requirements for mobile applications, in: 2009 24th International Symposium on Computer and Information Sciences. ISCIS, 555–560.

de Cleva Farto, G., Endo, A.T., 2017. . Reuse of model-based tests in mobile apps, in: Proceedings of the 31st Brazilian Symposium on Software Engineering - SBES'17. ACM Press, New York, New York, USA, pp. 184–193.

de Souza, L.S., Aquino Jr, G.S., de, 2014. MEFFORTMOB: A Effort Size Measurement for Mobile Application Development. Int. J. Softw. Eng. Appl. 5, 63–81.

Dewi, M.., Nur Atiqah Sia, A., 2015. Reviews on agile methods in mobile application development process, in: 2015 9th Malaysian Software Engineering Conference (MySEC). pp. 161–165.

Farooq, S.U., Quadri, S.M.K., Ahmad, N., 2011. Software measurements and metrics: role in effective software testing. Int. J. Eng. Sci. Technol. 3, 671–680.

Ferrucci, F., Gravino, C., Salza, P., Sarro, F., 2015. Investigating Functional and Code Size Measures for Mobile Applications, in: Proceedings - 41st Euromicro

Conference on Software Engineering and Advanced Applications, SEAA 2015. pp. 271–287.
Francese, R., Gravino, C., Risi, M., Scanniello, G., Tortora, G., 2015. On the Use of Requirements Measures to Predict Software Project and Product Measures in the Context of Android Mobile Apps: A Preliminary Study, in: Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015. pp. 357–364.
Franke, D., Kowalewski, S., Weise, C., Prakobkosol, N., 2012. Testing conformance of life cycle dependent properties of mobile applications, in: Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012. pp. 241 – 250.
Gao, J., Bai, X., Tsai, W.-T., Uehara, T., 2014. Mobile Application Testing: A Tutorial. Computer (Long. Beach. Calif). 2, pp. 46–55.
Giessmann, A., Stanoevska-Slabeva, K., de Visser, B., 2012. Mobile Enterprise Applications – Current State and Future Directions, in: 45th Hawaii International Conference on System Sciences. pp. 1363–1372.
Göth, B.R., 2015. Testing Techniques for Mobile Device Applications. Masaryk University.
Haoues, M., Sellami, A., Ben-Abdallah, H., 2017. A Rapid Measurement Procedure for Sizing Web and Mobile Applications Based on COSMIC FSM Method, in: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement. pp. 129–137.
Hauptmann, B., Junker, M., Eder, S., Amann, C., Vaas, R., 2014. An expert-based cost estimation model for system test execution, in: ACM International Conference Proceeding Series. pp. 159–163.
Heeringen, H. Van, Gorp, E. Van, 2014. Measure the functional size of a mobile app: Using the cosmic functional size measurement method, in: Proceedings - 2014 Joint Conference of the International Workshop on Software Measurement, IWSM 2014 and the International Conference on Software Process and Product Measurement, Mensura 2014. pp. 11–16.
Holl, K., Elberzhager, F., 2016. Quality Assurance of Mobile Applications : A Systematic Mapping Study, in: 15th International Conference on Mobile and Ubiquitous Multimedia ACM, New York, NY, USA,. pp. 101–113. https://doi.org/10.1145/3012709.3012718
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.
Jayakumar, K.R., Abran, A., 2013. A survey of software test estimation techniques. J. Softw. Eng. Appl. 6, 47–52.
Jin, C., Jin, S.W., 2016. Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization. Appl. Soft Comput. J. 40, 283–291.
Kamala Ramasubramani, J., 2016. ESTIMATION MODEL FOR SOFTWARE TESTING. ÉCOLE DE TECHNOLOGIE SUPÉRIEURE UNIVERSITÉ DU QUÉBEC.
Katherine, a.V., Alagarsamy, D.K., 2012. Conventional software testing vs cloud testing. Int. J. Sci. 3, 1–5.
Kaur, A., 2016. Review on agile approach to mobile application development. IJCAT – International J. Comput. Technol. 3, 200–203.
Kim, H., Choi, B., Wong, W.E., 2009. Performance testing of mobile applications at the unit test level, in: SSIRI 2009 – 3rd IEEE International Conference on Secure Software Integration Reliability Improvement. pp. 171–181.
Kim, H.K., 2012. Mobile applications software testing methodology, Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition. Communications in Computer and Information Science. Springer, Berlin, Heidelberg.
Kirubakaran, B., Karthikeyani, V., 2013. Mobile application testing — Challenges and solution approach through automation, in: 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering. pp. 79–84.
Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering.
Kushwaha, D.S., Misra, A.K., 2008. Software test effort estimation. ACM SIGSOFT Softw. Eng. Notes 33, 1–6.
Lazić, L., Mastorakis, N., 2009. The COTECOMO: COnstractive Test Effort COst MOdel, in: European Computing Conference. Lecture Notes in Electrical Engineering. Springer, Boston, MA.
Liu, Z., Hu, Z., Cai, L., 2014. Software Quality Testing Model for Mobile Application, in: MobiWIS 2014: Mobile Web Information Systems. Springer, Cham.
Lu, L., Hong, Y., Huang, Y., Su, K., Yan, Y., 2012. Activity page based functional test automation for android application, in: Proceedings of the 2012 3rd World Congress on Software Engineering, WCSE 2012. pp. 37–40.
Lusky, M., Powilat, C., Böhm, S., 2018. Software cost estimation for user-centered mobile app development in large enterprises, in: Advances in Human Factors, Software, and Systems Engineering. AHFE 2017. Advances in Intelligent Systems and Computing. Springer, Cham.
Amen, M., Mahmood, B.M., Lu, S., 2015. Mobile Application Testing Matrix and Challenges. In: Computer Science & Information Technology (CS & IT). Australia, Sydney, pp. 27–40.
Malavolta, I., Ruberto, S., Soru, T., Terragni, V., 2015. End Users' Perception of Hybrid Mobile Apps in the Google Play Store, in: Proceedings – 2015 in: IEEE 3rd International Conference on Mobile Services, MS 2015. New York, NY, USA, pp. 25–32.
Mendes, E., 2007. Cost Estimation Techniques for Web Projects, IGI Global.
Muccini, H., Francesco, A. di, Esposito, P., 2012. Software testing of mobile applications: Challenges and future research directions. 7th Int. Work. Autom. Softw. Test (AST 2012).
Nageswaran, S., 2001. Test Effort Estimation Using Use Case Points, In: Proceedings of 14th International Internet Software Quality Week.
Nguyen, V., Pham, V., Lam, V., 2013. qEstimation: a process for estimating size and effort of software testing, in: Proceedings of the 2013 International Conference on Software and System Process – ICSSP 2013. pp. 20–28.
Nidagundi, P., Novickis, L., 2017. New method for mobile application testing using lean canvas to improving the test strategy, in: 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT). pp. 171–174.
Nitze, A., 2013. Measuring Mobile Application Size Using COSMIC FP. DASMA Metr. Kongress 11, 101–114.
Nitze, A., Schmietendorf, A., Dumke, R., 2014. An analogy-based effort estimation approach for mobile application development projects, in: Proceedings - 2014 Joint Conference of the International Workshop on Software Measurement, IWSM 2014 and the International Conference on Software Process and Product Measurement, Mensura 2014.
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.
Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. 64, 1–18.
Petticrew, M., Roberts, H., 2006. Systematic Reviews in the Social Sciences. A Practical Guide. Blackwell Publishing, Systematic Reviews in the Social Sciences.
Preuss, T., 2013. Mobile Applications, Function Points and Cost Estimating, The IFPUG Guide to IT and Software Measurement, IFPUG. Auerbach Publications, Raton Boca, FL, USA.
Qi, K., Boehm, B.W., 2017. A light-weight incremental effort estimation model for use case driven projects, in: 2017 IEEE 28th Annual Software Technology Conference (STC). Gaithersburg, MD, pp. 1–8.
Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: 2008 Second International Conference on Research Challenges in Information Science. pp. 337–342.
Scharff, C., Verma, R., 2010. Scrum to Support Mobile Application Development Projects in a Just-in-time Learning Context, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering. pp. 25–31.
Sellami, A., Haoues, M., Ben-Abdallah, H., Abran, A., Lesterhuis, A., Symons, C., Trudel, S., 2016. Case Study Sizing Natural Language/UML Use Cases for Web and Mobile Applications using COSMIC FSM 1–42.
Shahwaiz, S.A., Malik, A.A., Sabahat, N., 2016. A parametric effort estimation model for mobile apps, in: 2016 19th International Multi-Topic Conference (INMIC). Islamabad, pp. 1–6.
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.
Silva, D.G.e., de Abreu, B.T., Jino, M., 2009. A Simple Approach for Estimation of Execution Effort of Functional Test Cases, in: 2009 International Conference on Software Testing Verification and Validation. IEEE, 289–298.
Soares, Y., Fagundes, R., 2017. Software time estimation using regression methods, in: IEEE Latin American Conference on Computational Intelligence (LA-CCI). Arequipa, pp. 1–6.
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.
Srivastava, P.R., 2015. Estimation of software testing effort using fuzzy multiple linear regression. Int. J. Softw. Eng. Technol. Appl. 1, 145.
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.
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.
Tunalı, V., 2014. Software Size Estimation Using Function Point Analysis – A Case Study for a Mobile Application, in: Mühendislik ve Teknoloji Sempozyumu (MTS7).
Umuhoza, E., Brambilla, M., 2016. Model driven development approaches for mobile applications: A survey, in: Younas M., Awan I., Kryvinska N., Strauss C., T.D. (Ed.), International Conference on Mobile Web and Information Systems 2016. Springer, Cham, pp. 93–107.
Usman, M., Mendes, E., Weidt, F., Britto, R., 2014. Effort estimation in Agile Software Development: A systematic literature review, in: Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE). ACM International Conference Proceeding. NY, USA, pp. 82–91.
Veenendaal, E., Dekkers, T., 1999. Test Point Analysis: A Method for Test Estimation, Project control for software quality : proceedings of the combined 10th European Software Control and Metrics conference and the 2nd SCOPE conference on software product evaluation, Herstmonceux. Shaker-Verlag, Maastricht, England.
Vilkomir, S., Amstutz, B., 2014. Using Combinatorial Approaches for Testing Mobile Applications, in: 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops. pp. 78–83.
Vogelezang, F., Ramasubramani, J., Rvamudhan, S., 2016. Modern Software Engineering Methodologies for Mobile and Cloud Environments, Modern Software Engineering Methodologies for Mobile and Cloud Environments,

Advances inSystems Analysis, Software Engineering, and High Performance Computing. IGI Global.
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.
Wasserman, A.I., 2010. Software engineering issues for mobile application development, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. ACM, 2010. pp. 397–400.
Zapata-Jaramillo, C.M., Torres-Ricaurte, D.M., 2014. Test Effort: a Pre-Conceptual-Schema-Based Representation. Dyna 81, 132–137.
Zein, S., Salleh, N., Grundy, J., 2016. A systematic mapping study of mobile application testing techniques. J. Syst. Softw. 117, 334–356.
Zein, S., Salleh, N., Grundy, J., 2015. Mobile application testing in industrial contexts: An exploratory multiple case-study. In: Fujita, H.G.G. (Ed.), Intelligent Software Methodologies, Tools and Techniques (SoMeT). Communications in Computer and Information Science, Springer, Cham.
Zhang, D., Adipat, B., 2005. Challenges, methodologies, and issues in the usability testing of mobile applications. Int. J. Hum. Comput. Interact. 18, 293–308.
Zhang, T., Gao, J., Cheng, J., Uehara, T., 2015. Compatibility Testing Service for Mobile Applications, in: Service-Oriented System Engineering (SOSE). IEEE Symposium, 179–186.
Zhu, X., Zhou, B., Hou, L., Chen, J., Chen, L., 2008a. An experience-based approach for test execution effort estimation, in: Proceedings of the 9th International Conference for Young Computer Scientists. ICYCS, 1193–1198.
Zhu, X., Zhou, B., Wang, F., Qu, Y., Chen, L., 2008b. Estimate test execution effort at an early stage: An empirical study, in: Proceedings of the 2008 International Conference on Cyberworlds, CW. IEEE Computer Society, 195–200.