International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 2
ISSN: 2454-4248
258 – 264

Cost Estimation Approach for Scrum Agile Software Process Model

Dr. Kiran Kumar Jogu
India Software Labs
IBM
Hyderabad, India
RajkiranJogu@gmail.com

Dr. K. Narendar Reddy
Dept. of CSE,
CVR College of Engineering,
Hyderabad, India
knreddy@cvr.ac.in

***Abstract*—** The software development in the industry is moving towards agile due to the advantages provided by the agile development process. Main advantages of agile software development process are: delivering high quality software in shorter intervals and embracing change. Testing is a vital activity in software development process model, for delivering a high quality software product. Often testing accounts for more project effort and time than any other software development activities. The software testing cost estimation is one of the most important managerial activities related to resource allocation, project planning and to control overall cost of the software development. Several models have been proposed by various authors to address the issue of effort and cost estimation. Most of the models are directly or indirectly depend on the source code of the software product. But, majority of the testing in software organizations is done in black-box environment, where the source code of the software is not available to the testing teams. In this paper, an alternative approach to software testing cost estimation for scrum agile software process model, by considering various testing activities involved in black-box testing environment is presented. The proposed approach is applied on four real world case studies and found that this approach provides more accurate estimation of the testing effort and cost, and helps the software managers in controlling the overrun of the project schedules and project costs.

Keywords*-Software industry; agile software development; scrum; black box testing environment; cost estimation approach.*

*****

I. INTRODUCTION

Traditional software development process models are being used for long time in software development. Present business demands the software products to be delivered in shorter intervals and software development environment having capability to embrace change at any stage of development. Traditional process models have difficulty in responding to change which often contributes success or failure of a software product [1]. Software requirements are dynamic which are driven by industry market forces. Agile approach to software development is suitable to such situations [2], [3]. Hence, more software companies are making a transition to agile software process models from traditional software development process models. Some of the key factors for success in an agile testing approach are: adopting an agile mindset, automating regression tests, collaborating and obtaining feedback from customer [4]. Some issues may arise when transition is made from traditional development to agile development. Common issues for agile models after migration from traditional models were identified in [5]. They are related to testing, test coverage, coordination overhead, and software release. In this paper we focused on testing related issues and testing cost estimation. Agile methods employ short iterative cycles, with prioritizing the requirements which actively involve users. Agile process models are iterative, incremental, self-organizing and emergent [6]. One of the agile process models which is being used in the software industry is "scrum". Scrum agile process model is defined in [7], [8]. In agile software development, testing is a vital activity for delivering a high quality software product to the customers. Often testing accounts for more project effort and time than any other software development activities. Since testing plays a major role in the success of the product, it is given a lot of importance in software development. Testing strategies for conventional process models are well established, but these strategies are not directly applicable to agile testing without modifications and changes. One of the important current research areas is the agile software testing strategies. It is widely accepted that the cost of software testing, debugging, and verification activities can range from 50 to 75 percent of the total software development cost in a typical commercial software development organization [9]. In most of the organization the software testing is done in the black-box software testing environment where the source code of the product is not accessible to the testing engineers. In black-box testing environment, the LOC information and function point information is not available. Therefore, there is a need for an approach to estimate effort & cost of software testing in black-box environment, which does not depend on source code, lines of code and number of function points.

In this paper, we propose an approach to cost estimation for back-box testing based on the testing activities involved in scrum agile development process. The testing activities are:Test plan preparation, Test suite development, Environment setup for testing, Verification of the fixed bugs which were reported in the previous testing cycle, Test suite execution, Test report generation, Test report analysis, and Reporting the bugs. This approach provides the estimated effort required for testing and estimated cost of the testing without depending on the source code of the product. In order to measure the correctness and capability of the proposed approach, we applied the proposed approach on four real world case studies. The results show that our approach can help the software testing managers in estimating the effort required for testing in each sprint, estimating cost of the software testing for each sprint and to minimize the overrun of the project schedules and testing cost.

This paper makes the following main contributions:

- We proposed a novel approach to cost estimation of software testing in scrum agile software developmentin black-box environment.
- We applied the proposed approach on four real world case studies which were developed in scrum agile process model.

The rest of the paper is organized as follows: Section II reviews the various software cost estimation models and summarizes related work. Section III describes details of testing in agile developmentand our work including an approach to cost estimation for black box software testing. Section IV describes the application of the proposed approach on four real world case studies and results obtained. Section V concludes and discusses future work.

II. Related Work

Software industry is transitioning to agile methodologies from traditional approaches. One of the popular agile process models which are being used in software companies is "scrum". Scrum main characteristic is, continuous deployment of working product increment after each sprint. As per the survey on agile methods given in [10], 54% of the software companies who are using agile methods are using Scrum. In the survey conducted by [11] on agile projects in different countries found that six critical factors contribute to agile project success. These factors are: agile software engineering techniques, customer involvement, project management process, team environment, team capability, and delivery strategy. One of the attributes related to the critical factor "agile software engineering techniques" is testing strategies. To address the above mentioned critical factor and its associated attribute, currently research is being carried out on agile testing strategies [12], [13].

Rani and Misra proposed a trade-off model between cost and reliability. This model estimates the testing cost of software for a desired level of reliability [14]. Vienneau, R.L proposed a model that can be used to estimate the cost of testing software if the release data is determined by some criteria, such as reliability. This model is based on underlying software reliability models and allows a manager to determine the optimum amount of time needed to test a software system [15]. A QC 3-D test model was introduced by Yuyu Yuan Shen Gu. This model builds a software test process with the cost control management and to make tradeoff between the quality and the cost [16]. Shimeall T.J. and Kelly T.J did research on isolating the costs involved in software testing and studied how the scheduling of software tests impacts the overall test costs. A formal model of software test costs was presented [17]. Chin-Yu Huang and Michael studied the impact of software testing effort & efficiency on the modeling of software reliability, including the cost for optimal release time. Thy presented two important issues in software reliability modeling & software reliability economics: testing effort, and efficiency. And proposed a generalized logistic testing-effort function that enjoys the advantage of relating work profile more directly to the natural flow of software development, and can be used to describe the possible testing-effort patterns. They also addressed the effects of new testing techniques or tools for increasing the efficiency of software testing. Based on the proposed software reliability model, they presented a software cost model to reflect the effectiveness of introducing new technologies. This method could help project managers determine when to stop testing for market release at the right time [18]. Benton proposed a systematic approach for estimating time and cost information in a software testing environment. This model is particularly well suited to solving complex, multivariable problems with time variant data. The model-based estimation system described in his paper was significantly less complicated than many formal estimation models (such as COCOMO II) [19].

Among the papers described above, most of the techniques and models assume that the source code of the software is available to estimate the cost of software testing in conventional software development process. But in most of the organizations the testing is performed in black-box environment where there is no accessibility to the source code. In this paper, for scrum agile software development process, an approach to software testing cost estimation for black-box testing environment, which does not depend on the source code is presented.

III. Agile Software Development Using Scrum

To provide consumers with continuous deployment of new features rapidly with the capability of embracing change at any stage of development, scrum is ideally suited for this purpose [7], [8], [20]. The scrum agile model is an iterative, incremental process of planning, development, testing, and deployment. In scrum at the end of each sprint a working increment is released and deployed. In XP (eXtreme Programming) at the end of an iteration, the working product may not be available. Hence, scrum leads to continuous deployment when compared to XP. Due to scrum's main characteristic of continuous deployment, software industry is transitioning to scrum agile software development. The scrum model is depicted in the Fig. 1 which is adopted from [7]. The model shown in Fig.1, is depicting the artifacts of their underlying activities. The main framework activities of the agile process model are: Creation of product backlog, Planning (Creation of sprint backlog and expanding the sprint backlog), and Sprint (consists of development activities). The scrum activities are performed by the scrum team which consists of product owner, development team, and scrum master. Product owner is responsible for creating and maintaining the requirements in product backlog. He/she creates stories for the requirements in the product backlog. Development team is responsible for developing the product by implementing the features in sprint backlog. The development team is cross functional. Cross functional means, team is responsible for design, development, testing, and deployment. The responsibility of the scrum master is to ensure that the scrum process is followed properly by the team.

The scrum activities lead to the following artifacts: product backlog, sprint backlog, task list to achieve sprint backlog, and working software product increment respectively. These artifacts are briefly discussed below.

Product backlog: The required product features or requirements identified by customer are added to product backlog. Features are prioritized as desired by the customer. The main source of agility in scrum model is the prioritized requirements list, which is flexible product backlog [8], [21]. Changes are inevitable. As the needs of the customers change the product backlog is continuously reprioritized. Hence, the software development is flexible. New features are selected from the backlog continuously and integrated and released as a working product increment at the end of the sprint. This means that one can deliver with increasingfunctionality morefrequently, which provides flexibility and the opportunity for adaptive planning [8].

 ISSN: 2454-4248
258 – 264

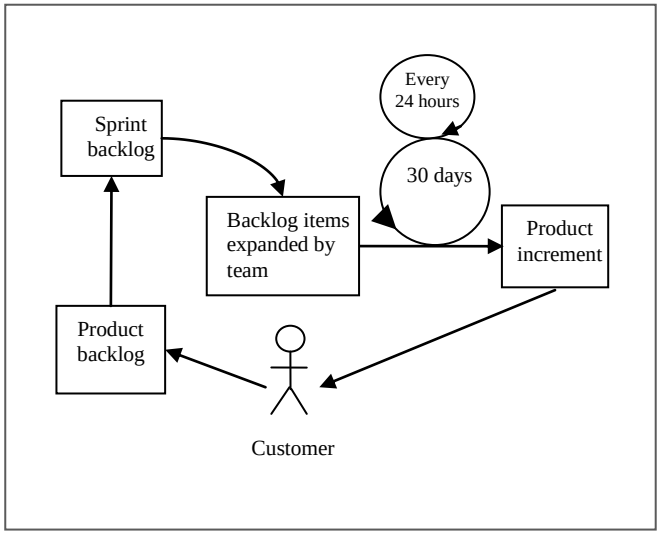

Figure 1. Scrum agile process model.

Sprint backlog: During first part of planning, product owner and development team together decides which features (user stories) will be part of the next sprint. The high priority features from product backlog are given preference. These features in this backlog are addressed during the sprint. Typical time-box for a sprint is 30 days. The changes (addition of new features) to the features in the ongoing sprint will not be accepted. But, changes (new features) can be added to the product backlog while the sprint is in progress.

Expanded sprint backlog: During second part of planning, development team analyses the user stories (features) in the sprint backlog and divides each user story in its tasks. These tasks are handled by different development team members during sprint.

Working software product increment: During sprint, development activities are carried out iteratively. Scrum meetings are held daily, typically of 15 minutes duration. Team discusses about the progress and what to be done in next 24 hours. At the end of sprint (30 days), working software product increment is delivered (deployed). Delivered product is evaluated by the customer to ensure that the features in the sprint backlog are implemented. Testing is important, because it is carried out to ensure the software product quality. Testing consumes most of the time during the sprint. A better software testing effort estimation system will help the software managers to resource allocation, project planning and to control overall cost of the software development. Hence, the authors of this paper focused on a novel approach to estimate the effort and cost of software testing. Testing activities and proposed testing method for scrum model are given in the following section.

A. Testing Activities in Scrum

Scrum is a framework for developing software products [22]. Various processes and techniques can be proposed and employed within the framework. Scrum framework specifies the following activities: planning (Creation of sprint backlog and expanding the sprint backlog), Sprint (consists of activities which can deliver a working software product increment implementing sprint backlog features in a given time-box(typically 30 days)). To propose a method for testing, first the sprint activities need to be considered and identified. One of the possible set of sprint activities can be eXtreme programming (XP) type development activities. The XP development activities could be: design, test driven development and refactoring, integration and regression testing, and validation testing before release. XP activities may not produce a working product increment after completing iteration(s) (in a given time-box). This may be because of the fact that this model is not based on predefined time-box based product release, hence the authors of this paper considered sprint activities which can deliver the working software product in predefined time-box. Sprint activities that are considered, are shown in Fig. 2. The activities are: design, development (coding), and testing. They are performed iteratively to produce a working product increment in a given time-box (sprint). The sprint activities are carried out iteratively to implement the features (user stories) in sprint backlog. The team for sprint contains scrum master and development team. Development team is cross-functional. They will be able to perform design, coding, and testing (unit testing and integration testing). Some of the development team members (testers) can be specifically meant for regression and functional testing. The responsibilities of the testers are: to plan and update test cases for sprint stories, automate test scripts if possible, execute the tests and report defects, and run regression tests and functional tests at the end of the sprint. Testers are also responsible for testing non-functional tests such as load testing and performance testing.

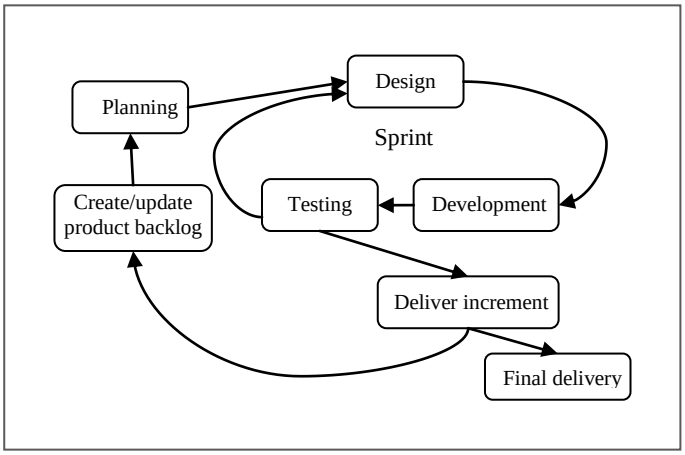

Figure 2. Sprint activities in scrum.

The testers in scrum agile software development participate in scrum ceremonies which includes sprint review, planning, daily and retrospective meetings. The testing activities for scrum model are depicted in Fig. 3. Testing strategy contains: unit testing, continuous integration, and regression testing which are carried out during the sprint. Whereas, functional and non-functional testing and user acceptance testing is carried out at the end of the sprint. The testing tasks during a sprint are incremental and iterative. Unit testing is done by the developer for finding the logical errors in a module. The bugs found in unit testing are debugged before integrating with other modules. Continuous integration is performed daily. Continuous integration enables to complete the increment in the scheduled sprint time. Regression testing is done after every integration test to ensure that newly integrated module has not introduced any new bugs. Functional test cases are created based on sprint backlog stories and executed at the end of the sprint.

 ISSN: 2454-4248
258 – 264

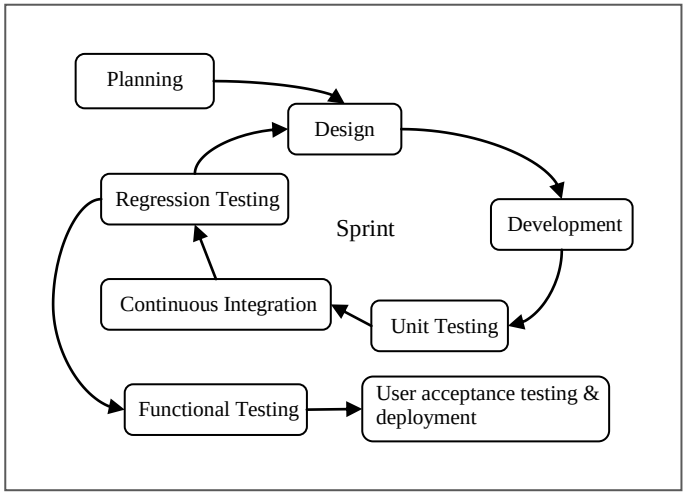

Figure 3. Testing activities in scrum.

Testing activities are automated. These testing tasks are conducted repeatedly and frequently, hence, automation will help to reduce the testing time. Since these tests are conducted iteratively on small number of features they increase the likely hood of finding bugs early in the project in intermediate releases (sprints) and in turn reduces the likely hood of magnifying and propagating the bugs to the final product. Because of this fact the quality of software product is better in agile software development. During deployment the product increment is tested by the user which is known as user acceptance testing (UAT) to ensure that all the user stories specified in the sprint backlog are actually implemented. In addition to testing functional requirements, it is essential to test non-functional requirements. Some of the typical non-functional requirements are: load testing, security testing, and performance testing. Tools are used for testing non-functional requirements. These non-functional tests are executed at the end of the sprint. Software testing automation is key for the agile testing. Irrespective of agile methodology, testing automation becomes the core of agile testing [13]. The purpose of software testing automation is to automate software testing activities. Manual testing is time consuming. Manual testing is not suitable for scrum agile testing where continuous deployment is required in shorter intervals. Moreover, since testing tasks are conducted iteratively during a sprint, through testing automation testing time can be reduced considerably. Tools are available to automate all the testing activities. With automation, testing efficiency can be improved and testing time can be reduced which enables to deploy the working product increments in shorter intervals.

B. The proposed cost estimation approach

The software testing activities consume about 50 to 60 percent of the overall software project resources and cost [23,24,25]. This indicates the need of a better software testing cost estimation model to estimate, plan and manage the resources in a better way.

In most of the software organizations, the testing is done in black-box environment, where the product code is not directly available to the test engineers and test managers. The cost estimation models which are depending on number of lines in the source code cannot be applicable for black-box testing environment. In this paper, we proposed an alternative way to estimate the testing cost, based on various testing activities involved in the black-box testing. The following are the major activities that are performed during the Black Box testing:

- Test Plan preparation (one time)
- Test Cases/test suite development (one time)
- Environment setup for testing (zero or more times)
- Verification of the fixed bugs which were reported in the previous testing cycle (zero or more times)
- Test Suite execution (more than one time)
- Test Report Generation (more than one time)
- Test Report Analysis (more than one time)
- Reporting the Bugs& Create/update product backlog (zero or more times)

In these various activities, the Test Plan preparation and the Test Cases/test suite development are one time activities that are performed in each sprint. The activities, verification of the fixed bugs which were reported in the previous testing cycle or sprint, and reporting the new bugs are performed zero or more times, depending on the bug fixes available in the current sprint and the bugs found during the current testing cycle of the sprint.

The test plan preparation (T_p) and the test suite development (T_s) are one time activities for a sprint. The effort required for these two activities is calculated in the following equation.

$$E_{init} = T_p + T_s \quad (1)$$

A product is developed in multiple sprints and each sprint produces a deliverable increment. The testing team receives an intermediate build for each of these sprints. In each and every sprintin the testing cycle the following activities need to be performed.

- Environment setup for testing (T_{env})
- Verification of the fixed bugs which were reported in the previous testing cycle (T_{bv})
- Test Suite execution (T_e)
- Test Report Generation (T_{rg})
- Test Report Analysis (T_{ra})
- Reporting Bugs & Create/update product backlog(T_{br})

As the above mentioned actives are performed on each and every build in a sprint, they occupies major portion of the overall software testing time. The time required to complete testing on an intermediate software build in a sprint is calculated using the following equation.

$$E_{ib} = T_{env} + \left((N_t \times T_e)\right) + T_{rg} + T_{ra} + T_{bv} + T_{br} \quad (2)$$

Where, the "T_e " indicates the average time required to execute a single test case and the "N_t " is the total number of the test cases executed for that particular testing cycle in that sprint.

The estimated effort required to complete the testing in all the sprints of the software product development is calculated as follows:

 ISSN: 2454-4248
258 – 264

$$Eib_{total} = Eib_1 + Eib_2 + \dots + Eib_n \quad (3)$$

where, 'n' indicates the number of builds in a test/development sprint.

$$E_{total} = E_{init} + Eib_{total} \quad (4)$$

The equation (4) gives the total estimated effort required to test theincremental product during one sprint, in man-hours. And, the total estimated testing cost of the software can be calculated using the following equation.

$$C_{total} = S_e \times E_{total} \quad (5)$$

where," S_e" is the average salary paid to a testing engineer per man-hour.

The salary paid to the employee per man-hour mainly depends on the organization and geography of the employee. So, the estimated testing cost for the product can be calculated based on these factors and using equation (5).

The following section describes the empirical evaluation of the proposed approach.

IV. CASE STUDIES AND RESULTS

The proposed approach is applied on four real-time ETL tool (Data ware housing tool) components have been developed in agile development process: Informix ETL Database (DB) Component, DB2 ETL DB Component, Teradata ETL DB Component and Oracle ETL DB Component. The Concepts explained in Figure. 4 are generic and applicable to all the above four test case studies. In Figure 4, ETL, which stands for "extract, transform and load", is the set of functions combined into one tool or solution that enables companies to "extract" data from numerous databases, applications and systems, "transform" it as appropriate, and "load" it into another databases, a data mart or a data warehouse for analysis, or send it along to another operational system to support a business process.

We have analyzed 32 completed software projects during 11 years of software industry experienceand estimated the average
time required for each activity that is involved in block-box testing. The average time for each activity is presented in the Table-I.

It can be found from the table that one time activity (for each sprint)"test plan preparation" takes maximum time. Total time for test suit development depends on number of test cases. Environment set up takes 1.5 hours, this activity is performed zero or more times. The total effort for a sprint depends on the effort required for different activities, number of test cases, number of builds, and number of bugs reported. Number of test cases depends on number of functionalities to be attained in an incremental product, in a sprint.

TABLE I. AVERAGE TIME REQUIRED FOR TESTING ACTIVITIES

Testing activity	Avg. Estimated effort
Test Plan preparation	2 Hrs
Test suite development	4 min / testcase
Environment setup for testing	1.5 Hrs
Verification of the fixed bugs	10 min / bug
Test Suite execution	1 min/test case
Test Report Generation	5 min
Test Report Analysis	15 min
Reporting the Bugs	14 min / bug

The empirical testing values given in table-I are used in different equations for estimating total effort for each sprint as part of incremental product development. Total effort for complete product depends on number of sprints.

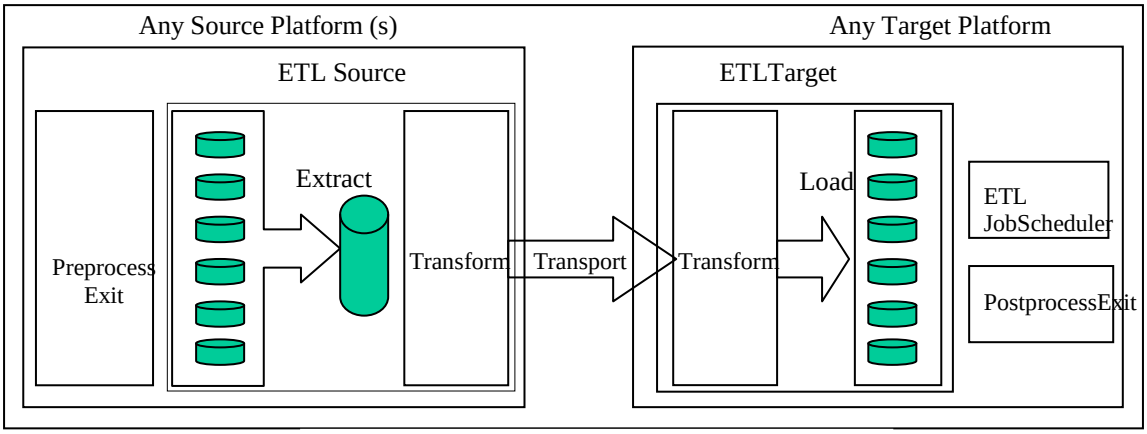

Figure 4. The ETL process

 ISSN: 2454-4248
258 – 264

A. *Effort estimation for Informix ETL Component*

The Informix ETL Component was developed in 9 sprints. The first column in the Table II gives the sprint details and the second column in the Table II gives the number of test cases executed in that sprint.

1) *Effort estimation for sprint 1:*

In sprint1, the initial effort for creating testing plan and test suite development for 'Informix ETL DB Component' is calculated using the equation (1):

E_{init} = 2 + (4 x 139)/60 = 11.26 Hrs

The estimated effort required to complete the testing on one build calculated using the equation (2) is:

Eib = (90+(139 x 1)+5+15+(4x10)+ (4x14))/60 = 5.75Hrs

From the historical data of the software testing organizations, the average number of intermediate builds that are provided to the testing team in a sprint is 2. The total effort required to complete the testing on all the intermediate builds of sprint is calculated using equation (3).

Eib_{total} = 5.75 + 5.75 = 11.5Hrs

The total effort required to test the incremental product during a sprint is calculated using equation (4).

E_{total} = 11.26 + 11.5 = 22.76Hrs

According to C. Jones [19] the average salary paid to a software engineer is $100 per hour. The total estimated cost for testing the incremental product is calculated using the equation (5).

C_{total} = 100 x 22.76 = 2276 $

The total effort and cost for each sprint are recorded in third and fourth column of the table II.

2) *Effort estimation for the remaining sprints:*

The effort estimation for the remaining sprints of the Informix ETL component is calculated as shown in the above sub section and recorded the results in third and fourth column of the table II.

So, the total estimated cost to test the 'Informix ETL DB Component' project for all the sprints is 23398$. In this paper we have taken $100 per hour as average salary paid to a software engineer. However, the average salary paid to a software engineer varies based on the organization and the geography location. Using the proposed equations, project manager can estimate the amount of effort and cost required for the testing activities during the software product development.

B. *Effort estimation for the remaining three case studies*

For the remaining three projects the testing effort and the testing cost are estimated using the proposed approach as explained in above two subsections and the final results are given in the Table III.

TABLE II. INFORMIX ETL COMPONENT TEST DETAILS

Sprints Si	N_t	E_{total} (Hrs)	C_{total} ($)
1	139	22.76	2276
2	185	27.36	2736
3	170	25.86	2586
4	200	28.86	2886
5	185	27.36	2736
6	123	21.16	2116
7	154	24.20	2420
8	185	27.36	2736
9	202	29.06	2906

TABLE III. CASE STUDIES AND RESULTS

Project	Sprints *Sn*	N_t	E_{total} (Hrs)	C_{total} ($)
Informix ETL DB Component	9	1543	233.98	23398
DB2 ETL DB Component	7	1039	112.76	11276
Teradata ETL DB Component	8	1290	137.87	13787
Oracle ETL DB Component	12	1958	276.79	27679

V. CONCLUSIONS

This paper describes various testing activities in the scrum agile development process and also presents an approach to software testing cost estimation in black-box environment, which is independent of the lines of code available in a software product. This approach is easy to apply by the practitioners in the software organizations, and provides more accurate testing cost and testing effort estimations.

The proposed equations, empirical testing data, and proposed approach are applied on four case studies successfully. The four real world ETL tool software components that are used by most of the organizations are the case studies for this paper. The four case studies are:Informix ETL DB Component, DB2 ETL DB Component, Teradata ETL DB Component and Oracle ETL Component. The detailed analysis results are given for Informix ETL DB Component, where for the other three case studies final results are given.

The estimation of testing effort and cost based on the proposed approach in four case studies enabled project manager to plan, allocate resources, and to complete the projects on schedule and within the estimated testing budget successfully without schedule and budget overruns.

As part of future work, the proposed approach needs to be applied on more case studies from different domains. This enables to refine the approach if required, based on new research findings.

 ISSN: 2454-4248
258 – 264

REFERENCES

[1] L.Williams and A.Cockburn,(2003) "Agile software development: it's about feedback and change", IEEE Computer, 36(6), pp. 39-43.

[2] J.Highsmith and A.Cockburn, (2001) "Agile software development: The business of innovation", IEEE Computer, 34(9), pp-120-127.

[3] C.R.Jakobsen and J.Sutherland, (2009) "Scrum and CMMI going from good to great", Agile Conference (AGILE), pp. 333-337.

[4] L.Crispin and J.Gregory, (2009) "Agile testing: A practical guide for testers and agile teams", Addison-Wesley.

[5] Kai Petersen and Claes Wohlin, (2010) "The effect of moving from a plan-driven to an incremental software development approach with agile practices: an industrial case study", Empirical Software Engineering, 15(6), pp.654-693.

[6] B.Boehm and R.Turner, (2005) "Management challenges to implementing agile processes in traditional development organizations", IEEE Software, 22(5), pp.30-39.

[7] K.Schwaber and M.Beedle, (2002), Agile software development with scrum, Prentice Hall.

[8] K.Schwaber, (2004), Agile project management with scrum, Microsoft Press.

[9] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification", IBM Systems Journal, vol. 41, no. 1, pp.4-12,2002.

[10] VersionOne, (2013) "7th Annual State of Agile Development Survey". Retrieved in November 2013 from http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf.

[11] T.Chow and D.Cao, (2007) "A survey study of critical success factors in agile software projects", The Journal of Systems and Software, pp. 961-971.

[12] Theodore D. Hellmann, Abhishek Sharma, Jennifer Ferreira, and Frank Maurer, (2012) "Agile testing: past, present, and future - charting a systematic map of testing in agile software development", Agile Conference (AGILE), pp. 55-63.

[13] E.Collins, A.Dias-Neto, and V.F.de Lucena, (2012) "Strategies for agile software testing automation: An industrial experience", IEEE 36th Annual Computer Software and Applications Conference Workshops (COMPSACW), pp. 440-445.

[14] Rani and R. B. Msra, "On Determining the Software Testing Cost to Assure Desired Field Reliability", Proceedings of the IEEE INDICON, Page(s): 517 - 520, 2004.

[15] Vienneau, R.L., "The cost of testing software ", Reliability and Maintainability Symposium, Page(s): 423 - 427, 1991.

[16] Yuyu Yuan, Shen Gu, "Research and Establishment of Quality Cost Oriented Software Testing Model", IEEE CCECE/CCGEI, Ottawa, Page(s): 2410 - 2415,May 2006.

[17] Timothy J. Shimeall,Timothy J. Kelly, "Examination of a Software-Test Cost Model",Reliability and Maintainability Symposium, Page(s): 26 - 30,1994.

[18] Chin-Yu Huang and Lyu M.R., "Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency",IEEE Transactions on Reliability,Page(s): 583 - 591, 2005.

[19] C. Jones. "Applied Software Measurement: Assuring productivity and quality", McGraw-Hill, 1997.

[20] Puneet Agarwal,(2011) "Continuous SCRUM: Agile management of SAAS products", ISEC' 11: Proceedings of the 4th India Software Engineering Conference, February 2011.

[21] K.Lukasiewicz and J.Miler, (2012) "Improving agility and discipline of software development with the Scrum and CMMI", IET Software, pp. 416-422.

[22] K.Schwaber and J.Sutherland, (2013) "The scrum guide: The definitive guide to scrum: The rules of the game", https://www/scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100.

[23] Boehm. B.W., "Software Engineering Economics", Prentice Hall,1981.

[24] Xic. M., "On the determination of optimum software release time", Proceedings of 1991 International Symposium on Software Reliability Engineering. Austin, Texas. pp 218-224. 17-18 May 1991

[25] Yamada. S, and Osaki.S, "Optimal Software release polocies with simulatneuous cost and reliabiltiy requirements", European Journal of Operational Research. 31:46-51, 1987