Early System Test Effort Estimation Automation for Object-Oriented Systems

Pulak Sahoo, J. R. Mohanty and Debabrata Sahoo

Abstract Quality and reliability are the two most important criterions used for judging a software product from customer's point of view. Software testing plays a critical role in delivering a high-quality software product. An early estimation of system test effort enables software organizations to plan and execute required test activities thoroughly. This results in the product meeting the required quality goals and improving customer acceptability. In this work, we propose a method for prediction of system test effort from Use Case models created in Requirement Analysis phase of software development. The estimation process includes automation of steps for extracting parameters from the system's Use Case models required for estimation of system test effort.

Keywords Use Case model · Use Case · Actor · Unified modeling language Test effort estimator · CASE tool · UCPM

1 Introduction

Quality of a software product is a major concern for both customers and the organizations developing it. It is well known that software testing [1] is essential for producing quality software products. For conducting a thorough and effective testing, test planning is very much essential. Test planning based on early test effort estimations is an area of attention for software development organizations of late.

P. Sahoo (✉)
School of Computer Engineering, KIIT University, Bhubaneswar 751024, India
e-mail: sahoo_pulak@yahoo.com

J. R. Mohanty
School of Computer Applications, KIIT University, Bhubaneswar 751024, India
e-mail: jnyana1@gmail.com

D. Sahoo
Utkal University, Bhubaneswar 751004, India
e-mail: debabratasahoo@live.com

S. C. Satapathy et al. (eds.), *Information and Decision Sciences*,
Advances in Intelligent Systems and Computing 701,
https://doi.org/10.1007/978-981-10-7563-6_34

If the effort required for testing a system is known early in Requirement Analysis phase, it will enable the project team to plan for the schedule and resources early. This will in turn help with effective conduction of test activities within required timeline.

Unified Modeling Language is currently the most popular standard for systems developed using object-oriented methodology [2]. Use Case models are created to capture the functional requirements, user interfaces, and scope of the system in the Requirement Analysis phase. Use Case models contain elements such as Use Cases, Actors, and their Associations. Since Use Case models are available early, these models can be used to provide inputs for an initial estimation of system test effort.

Our work proposes an estimation method for system test effort using Use Case models of the system. An automated parsing tool has been developed to extract essential information from Use Case models of the system. The Use Case models are created using a CASE tool called ArgoUML. This information is stored in a repository to be used for system test effort estimation.

ArgoUML is an open source Unified Modeling Language CASE tool written in Java. It supports exporting of Use Case models and essential parameters to XML metadata interchange (XMI) format, which then serves as inputs to the parsing tool.

The remainder of this paper is structured as follows. In Sect. 2 titled Related Work, we have described a number of relevant estimation approaches. Section 3, Proposed Method, contains the Use Case model-based test effort estimation method proposed by us. In Sect. 4, Implementation and Experimental Study, we describe the architecture, implementation steps, and experiments. Section 5, Conclusions and Future Work, provides the summary of our work and discusses the future scope.

2 Related Work

Over the years, experts have proposed a number of methods to estimate development and test effort for software products. In this section, we have discussed some relevant estimation methods which are widely used by software projects.

Function Point Analysis (FPA) method by Albrecht [3] is widely used to estimate the size of software products in terms of function points based on functionalities offered. In 1996, Caper Jones [4] proposed a method to calculate approximate number of acceptance test cases from function points using below formula.

$$\text{Number of Test Cases} = (\text{Function Points})^{1.2} \quad (1)$$

In 2000, Boehm proposed COCOMO [5] model, which estimates software development effort from the size of the system expressed in lines of code (LOC). Proposed in 2000, the Test Point Analysis (TPA) [6] method can estimate acceptance test effort of a system in test points based on its size in function.

Although FPA-based methods can provide early estimations, collecting the detailed inputs required for this process is time taking and can sometimes be costly.

The Use Case Points Methodology (UCPM) [7] was introduced to estimate system development effort from its Use Cases. The effort is expressed in Use Case Points calculated based on number of actors, number of transactions, and technical and environmental factors. In 2001, paper [8] proposed a refined UCPM-based method to estimate acceptance test effort. This method identified and classified Actors and Use Cases to assign points resulting in a total unadjusted Use Case Points for the system. Adjusted Use Case Points were obtained by taking into account nine technical and environmental factors. In paper [9], a more refined approach called N-Weighted method was proposed to estimate test activity effort based on systems Use Cases. This method separated Use Case scenarios into two types: normal and exceptional. Since a normal scenario has more steps than an exceptional scenario, the higher score was assigned to it.

In 2011, Paper [10] compared the accuracies of various UCPM-based estimation methods. It proposed a reduction in a number of environmental and technical factors from 21 to 6 by conducting factor analysis through experts. It also changed the Use Case Points calculation process by counting Use Case steps instead of transactions. But this method estimates development effort, not test effort. The estimation accuracy depends on expert's opinion and historical data of the organization.

In 2007, Paper [11] proposed an innovative approach to produce test cases from a system's Use Case and Sequence models. This method translated Use Case models to Use Case model Graph and Sequence models to Sequence model Graph. Both graphs were combined to create System Testing Graph which was traversed to produce test cases. Similarly, in 2010, Paper [12] proposed a method to produce test cases from State-Chart and Activity models. This method produced a combined state activity model and traversed all the basis paths to generate test cases. The abovementioned approaches can be expanded to produce test effort by classifying each test case by complexity and assigning scores. Unavailability of sequence, state-chart, and activity models early proves to be the hindrance for early estimation of test effort.

From above described methods, it is clear that there is a need for developing a simple and automated approach for estimating early test effort with reasonable accuracy. In this work, we propose an early system test effort estimation method using Use Case models and automation for extracting essential information from Use Case models required for the estimation process.

3 Proposed Method

For early system test effort estimation, we propose to take Use Case models of the system as inputs. Use Case models are the first UML model created in the requirement stage to capture functional requirements and user interfaces of

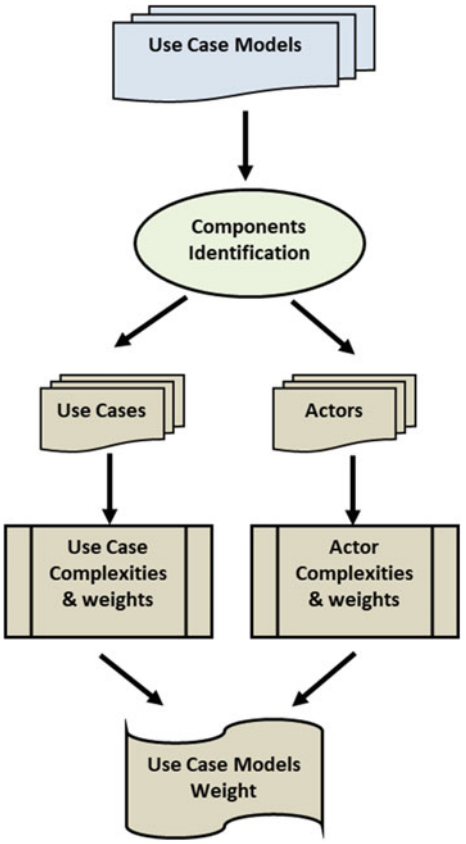

Fig. 1 Steps to determine Use Case model weights

the system. Use Case models have components like Use Cases, Actors, and Associations between those [13]. Shown in Fig. 1 is the diagrammatic view of steps involved in determining Use Case model weights based on the complexity of the involved components.

3.1 Use Case and Actor Complexity

Use Cases present in Use Case models represent functional units of the system. A Use Case contains Use Case scenarios [14] of types main or exceptional. While the main scenario is made up of primary steps to achieve some functionality, the exceptional scenario contains steps for error handling. The scenarios include transactions. The complexity of a Use Case from testing perspective is calculated by counting the transactions in it. Weights assigned to normal scenario are higher than that assigned to exceptional scenarios due to the presence of greater number of checks. Apart from this, a Use Case's complexity depends on number of interacting Actors.

$$\text{Usecase\_wt} = F1(\text{normal\_scenarios}, \text{excep\_scenarios}, \text{int\_actors}) \quad (2)$$

In Use Case models, Actors interacts with Use Cases to achieve some functionality. Actor may be an external user or an interfacing system. Actor's complexity depends on the mode of interaction with Use Cases. An Actor interacts with Use Cases via GUI, API, Protocol-Driven Interface, or Data Store. Apart from this, Actor complexity depends on number of interacting Use Cases.

$$\text{Actor\_wt} = F2(\text{comm\_type}, \text{int\_usecases}) \quad (3)$$

3.2 Use Case Model Weight

To calculate Use Case model weight, we combine the Use Case and Actor weights obtained from Eqs. 2 and 3. A system may contain a number of Use Case models. Summing up weights for all Use Case models results in the total weight of the system.

$$\text{Use Case Model\_wt} = F1() + F2() \quad (4)$$

After computing total weight of the system, adjustments will be applied by factoring in organization-specific technical and environmental factors relevant to testing of the software product. To this adjusted weight, organization-specific productivity factor will be applied to get system test effort.

4 Implementation and Experimental Study

The proposed architecture of UML test effort estimator containing steps of implementation is shown in Fig. 2. The major components are as follows: ArgoUML CASE tool, Use Case model Parser, Use Case model Classifier, Use Case model repository, test effort estimator, and classification setup interface. The estimator will produce Unadjusted Test effort, which will be adjusted by applying technical and environmental factors specific to the system and organization.

In the first step, Use Case models of the system are created with sufficient details by project team using ArgoUML CASE tool. The models are then exported to XMI format. In the second step, the Use Case model Parser extracts component information like Use Cases, Actors, and their associations from the XMI files and stores them in Use Case model repository. In the third step, the Use Case model component classifier categorizes the components based on complexity classification setup defined by subject matter experts. In step four, the test effort estimator computes the Unadjusted Test effort for the Use Case models. Later, relevant

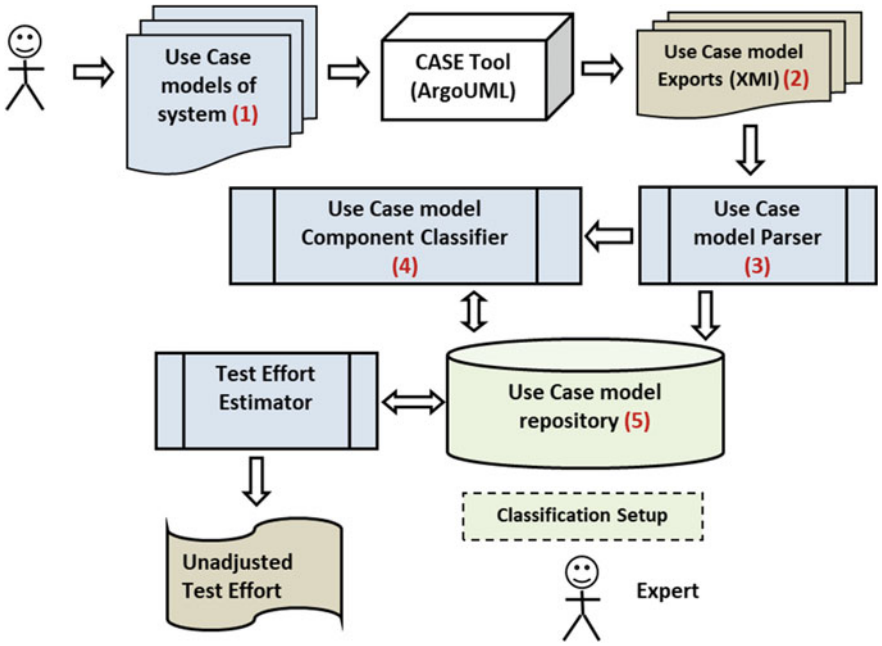

Fig. 2 Use Case model estimator architecture

technical and environmental factors can be applied to compute Adjusted Test efforts. In this work, we have implemented the first two steps on a real project. The project is titled "Cluster-based Agricultural Benefit Allocation (CABA)" which was executed by a reputed national IT organization. A brief description of CABA is given below.

For accelerated crop production, large clusters of agricultural lands are taken up for providing benefits (crop seeds) to farmers based on a cropping system. Using this system, the Admin allocates benefits to various districts according to crop production capacity. Then, the DDA (Deputy Director of Agriculture) distributes the allocated benefits among blocks and selects the crop varieties for production. Following this, the AAO (Assistant Agriculture Officer) selects the clusters and corresponding VAWs (Village Agriculture Worker) of that area to distribute the benefits. VAWs provide the seed requirements of the cluster to the registered seed agencies based on the production of crops. The seed agencies provide seeds to farmers according to the requirements and VAWs record the distribution of benefits among the clusters.

The Use Case models for CABA were created using ArgoUML and exported to XMI formats. Figure 2 shows the Use Case models for Crop Benefit Target Allocation (CBTA) and Crop Benefit Target Distribution (CBTD) modules. Figure 3 shows the Use Case model for Crop Benefit Cluster Distribution (CBCD) model and the XMI export file (Fig. 4).

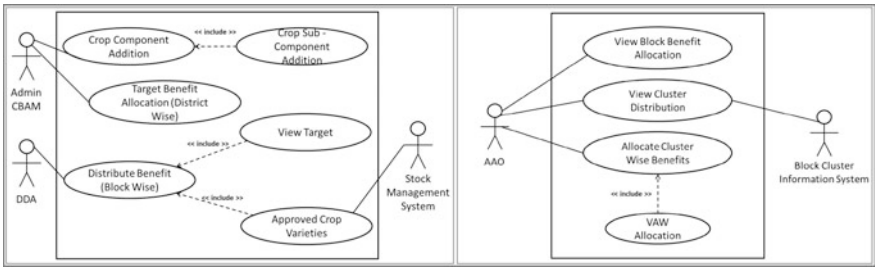

Fig. 3 Use Case models for CBTA and CBTD modules

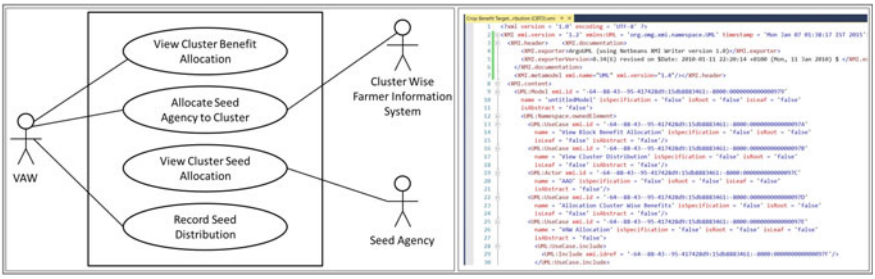

Fig. 4 Use Case model for CBCD module and XMI export

Module_ID	Use_Case_Model	Use_Case	No_of_Normal_Transactions	No_of_exception_Transactions	No_of_Interacting_Actors
CABA	CBTA	Crop_Component_addition	10	3	1
CABA	CBTA	Crop_Sub_Component_addition	8	2	0
CABA	CBTA	Target_Benefit_Allocation	14	3	1
CABA	CBTA	Distribute_Benefit	15	2	1
CABA	CBTA	View_Target	5	1	0
CABA	CBTA	Approved_Crop_Varteities	5	1	1
CABA	CBTD	View_block_Benefit_allocation	6	1	1
CABA	CBTD	View_Cluster_Distribution	8	2	2
CABA	CBTD	Allocate_Clusterwise_Benefits	15	3	1
CABA	CBTD	VAW_Allocation	7	1	0
CABA	CBCD	View_Cluster_Benefit_Allocation	6	1	1
CABA	CBCD	Allocate_Seed_Agency_to_Cluster	9	2	2
CABA	CBCD	View_Cluster_Seed_Allocation	7	1	1
CABA	CBCD	Record_Seed_Distribution	6	1	1

Fig. 5 Use Case model—Use Case data in repository

Module_ID	Use_Case_Model	Actor	Communication Type	No_of_Interacting_usecases
CABA	CBTA	Admin_CBAM	GUI	2
CABA	CBTA	DDA	GUI	1
CABA	CBTA	Stock_Management_System	Data Store	1
CABA	CBTD	AAO	GUI	3
CABA	CBTD	Block_Cluster_Information_system	Data Store	1
CABA	CBCD	VAW	GUI	3
CABA	CBCD	Clusterwise_farmer_Information_System	Data Store	2
CABA	CBCD	Seed_Agency	GUI	1

Fig. 6 Use Case model—actor data in repository

In the second step, the Use Case model Parser extracts relevant component information from Use Case models. It extracts all the Use Cases present in the models along with number of scenario transactions (both normal and exceptional) and number of interacting actors for each Use Case. It also extracts all the Actors present in the models along with the communication types and number of interacting Use Cases. This information is then stored in Use Case model repository to be used later for carrying out the remaining estimation steps. The two main data stores of Use Case model repository are shown in Figs. 5 and 6.

5 Conclusions and Future Work

In this work, we have proposed a simple method for system test effort estimation for object-oriented systems using Use Case models in early stage of software development. An early estimation of test effort will help project teams to plan ahead for the system testing phase, so that a quality product is delivered within the timeline. The estimation method and its steps are explained along with the estimator architecture. We have implemented the steps to export Use Case models created using ArgoUML CASE tool into XMI format and extracting relevant component information from Use Case models using a Use Case model parser. The extracted details are stored in Use Case model repository and will be used to estimate system test effort required for the system. An experimental study was conducted on a real project named Cluster-based Agricultural Benefit Allocation (CABA) executed by a reputed IT organization.

References

1. Jorgensen, P.C.: Software testing: a craftsman's approach. CRC press (2016)
2. Binder, R.V.: Testing object-oriented systems: models, patterns, and tools. Addison-Wesley Professional (2000)
3. Albrecht, A.: Measuring application development productivity. Proc. Joint SHARE/GUIDE/IBM Appl. Develop. Symp. **10**, 83–92 (1979)
4. Capers, J.: Applied software measurement. McGraw-Hill (1996)
5. Boehm, B., Horowitz, E., Madachy, R., Reifer, D., Clark, B., Steece, B., Brown, W., Chulani, S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice Hall (2000)
6. Van Veenendaal, E.P.W.M., Dekkers, T.: Test point analysis: a method for test estimation (1999)
7. Karner, G.: Metrics for objectory. Diploma Thesis. University of Linköping, Sweden (1993)
8. Nageswaran, S.: Test effort estimation using use case points, pp. 1–6. Quality Week (2001)
9. de Almeida, É.R.C., de Abreu, B.T., Moraes R.: An alternative approach to test effort estimation based on use cases. In: International Conference on Software Testing Verification and Validation, pp. 279–288. IEEE (2009)
10. Ochodek, M., Nawrocki, J., Kwarciak, K.: Simplifying effort estimation based on Use Case points. In: Information and Software Technology, vol. 53, pp. 200–213. Elsevier (2011)
11. Sarma, M., Mall, R.: Automatic test case generation from UML models. In: 10th International Conference on Information Technology, pp. 196–201. ICIT 2007, IEEE (2007)
12. Swain, S.K., Mohapatra, D.P., Mall, R.: Test case generation based on state and activity models. J. Obj. Technol. **9**, 1–27 (2010)
13. Sahoo, P., Mohanty, J.R.: Early test effort prediction using UML diagrams. Indones. J. Electr. Eng. Comput. Sci. **5**, 220–228 (2017)
14. Hussain, A., Nadeem, A., Ikram, M.T.: Review on formalizing use cases and scenarios: scenario based testing. In: International Conference on Emerging Technologies (ICET), pp. 1–6. IEEE (2015)