2017 43rd Euromicro Conference on Software Engineering and Advanced Applications

Towards Execution Time Prediction for Manual Test Cases from Test Specification

Sahar Tahvili*†, Mehrdad Saadatmand*, Markus Bohlin*, Wasif Afzal†, Sharvathul Hasan Ameerjan†

*Research Institutes of Sweden (RISE) RISE ICT/SICS Västerås, Sweden
†Mälardalen University, Västerås, Sweden
Email: {sahar.tahvili, mehrdad.saadatmand, markus.bohlin}@ri.se
† wasif.afzal@mdh.se, san15014@student.mdh.se

***Abstract*—Knowing the execution time of test cases is important to perform test scheduling, prioritization and progress monitoring. This work in progress paper presents a novel approach for predicting the execution time of test cases based on test specifications and available historical data on previously executed test cases. Our approach works by extracting timing information (measured and maximum execution time) for various steps in manual test cases. This information is then used to estimate the maximum time for test steps that have not previously been executed, but for which textual specifications exist. As part of our approach, natural language parsing of the specifications is performed to identify word combinations to check whether existing timing information on various test activities is already available or not. Finally, linear regression is used to predict the actual execution time for test cases. A proof-of-concept use case at Bombardier Transportation serves to evaluate the proposed approach.**

***Index Terms*—Software Testing, Execution time, Linear Regression, NLP, Optimization, Test Specification, Estimation**

1. Introduction

The main objective of software testing is detecting as many critical bugs in the system under test as possible. Since software testing is a time consuming and a costly process, it is possible that the planned time and budget for testing are not sufficient for executing all designed test cases [1]. In such a case, we need to prioritize test cases ensuring that the most important test cases are executed first.

The criteria for test case selection and prioritization are dependent on many factors such as the application, requirements and customers' perspective. In our previous work, some of the important criteria used in scheduling the order of test execution are analyzed [2], [3]. One such criterion is the test case execution time, which can enable efficient usage of testing resources [4]. The test case execution time is one factor contributing to the overall software test effort [5]. Through knowing the execution time for each test case, we can measure the required time for testing a system [6].

Since execution time measurement is not an easy task, a number of assumptions need to be made first. Most of the existing methods focus only on automated testing via analysis of executable code, in order to construct predictors or select important features [6]. However, the lack of automated testing in some contexts, such as when to test certain aspects of a system (e.g., safety), forces tester to perform manual testing [7].

In this work in progress paper, we propose an approach for measuring the required execution time for manual test cases. We have structured our approach into two main terminologies of execution time: maximum and actual execution time. Thereby, two algorithms are designed: *estimation* and *prediction* algorithms which will compute maximum and actual execution time respectively. The *estimation* algorithm takes test specifications written in natural language and identifies key elements, such as verbs and objective arguments, and stores them in a database. By analyzing the test records and matching the set of extracted elements against previously executed test steps of test cases, an execution time can be assigned to each activity in the test specification. In case there is no match, a baseline time which represents the response time that a particular system takes to react to a given input, is assigned. The *prediction* algorithm is based on linear regression to predict the required execution time for newly created test cases.

The remainder of this paper is organized as follow: Section 2 presents background and motivation of the initial problem, Section 3 describes the proposed approach. A proof of concept has been summarized in Section 4 through analyzing an industrial test example and finally Section 5 concludes the paper.

2. Background and Motivation

A manual test case generally consists of several steps written in a natural language. These steps describe actions, input data and the expected reactions (output) from the object under test, and checks if a particular test condition is satisfied or not. Before the steps of a test case can start, there are also, in general, certain pre-conditions that need to be met to put the system in a testable state.

Moreover, to allow the system to react, it is common that waiting activities are specified. We therefore assume a generic model of test cases with four different elements: pre-conditions, waiting times, actions and reactions.

TABLE 1: A test specification example from the safety-critical train control management system.

Initial state: No active cab		
Step	Action	Reaction
1	Login at the IDU as " Maintainer not driving" in A_1 cab	Check that: Cab A_1, MIO-S " Head light half-beam on left=FALSE" Cab A_2: MIO-S " Head light half-beam on right=FALSE"
2	Log out in cab by removing key card in cab A_1 Login at the IDU as driver in A_1 cab	Check that: Cab A_1, MIO-S " Head light half-beam on left=TRUE" Cab A_2, MIO-S " Head light half-beam on right=TRUE"
3	Log out in cab removing key card in cab A_1	Check that: MIO-S " Head light half-beam on left=TRUE" MIO-S " Head light half-beam on right=TRUE"
4	Wait 20 seconds	

Given the above context, we investigate the following research question:

RQ: *How to compute the test case execution time based on different elements of a manually written test case?*

Table 1 shows a part of a real, industrial test specification from a safety-critical train control management system, where the description of steps are outlined. The required time for performing some of the activities in Table 1 is available before execution, for instance, the waiting time in Step 4 of the test case (Table 1).

A test specification can also have several properties such as:

- **Test case size:** total number of test steps, which describe the execution sequence. Each test step consists of an action and a reaction. Each step is marked as either pass or fail based on the comparison between the expected and actual outcome [8].
- **Total activities:** a test step consists of at least one test activity, which represents the goal of the test step. The total test activities for a test case can be denoted as a_t.
- **Total waiting:** a wait for time step enables a test case to pause for a specified time before continuing execution. This helps synchronize multiple asynchronous calls in a single test case. It also enables testers to test more complex scenarios, such as those involving a long-running process that requires multiple interactions [8]. The total waiting time for a test case is simply the sum of waiting times written as test steps, denoted as w_t.
- **Total pre-conditions:** the total number of pre-conditions in a test case is denoted by c_t.

Let $t_1, t_2, \dots, t_n$ be the required time for executing a_t, w_t and c_t, then the total execution time (T_n) for a test case TC (which has n steps) of a given size is:

$$T_n = \sum_{j=1}^{n} t_j \tag{1}$$

Since the required time for performing the above mentioned activities is variable, we propose an *estimation* algorithm (Section 3.1) for solving it.

3. The Proposed Approach

This section describes our proposed approach, based on two main algorithms, for computing execution time for manual test cases. To achieve this target, several sources such as test specification, test script and test records will be analyzed. Before describing the proposed algorithms, we define our concept of test execution time as two main terminologies:

- **Maximum Execution Time (MT):** is the maximum time for testing an object successfully. In other words, MT is the maximum time that a test step (or test case) is allowed to take for testing an object. Suppose the maximum execution time for a test step is t seconds; if the system takes more time than t for testing an object, testing process need to be stopped and the test step is considered as a failure. MT is independent from the system properties. Automatic generation of test scripts from test specifications is another place where the concept of MT is useful. Thus, the testers are able to assign the upper boundary (test item timeout) of the required time for execution per test step.
- **Actual Execution Time (AT):** is the real time (t') taken by the system executing a test activity, which is dependent on the system properties. We need to consider that the actual execution time per test activity is equal or less than maximum execution time ($t' \leq t$).

Moreover, we introduce m_t as the system baseline reaction time, which represents the response time that system unit takes to react to a given input. The baseline time would be assigned as a required execution time for the steps of newly designed test cases which have never been executed before and there exists no similar and matching activity for the steps of such test cases. By measuring AT in several systems, we have this chance to have a better value for m_t.

The remainder of this section explains the structures of *estimation* and *prediction* algorithms which will compute maximum and actual execution time respectively. Since AT is a machine dependent parameter, we utilize the *prediction*, as a suitable term for the proposed algorithm. Furthermore, an overview of the proposed approach has been illustrated in Figure 1.

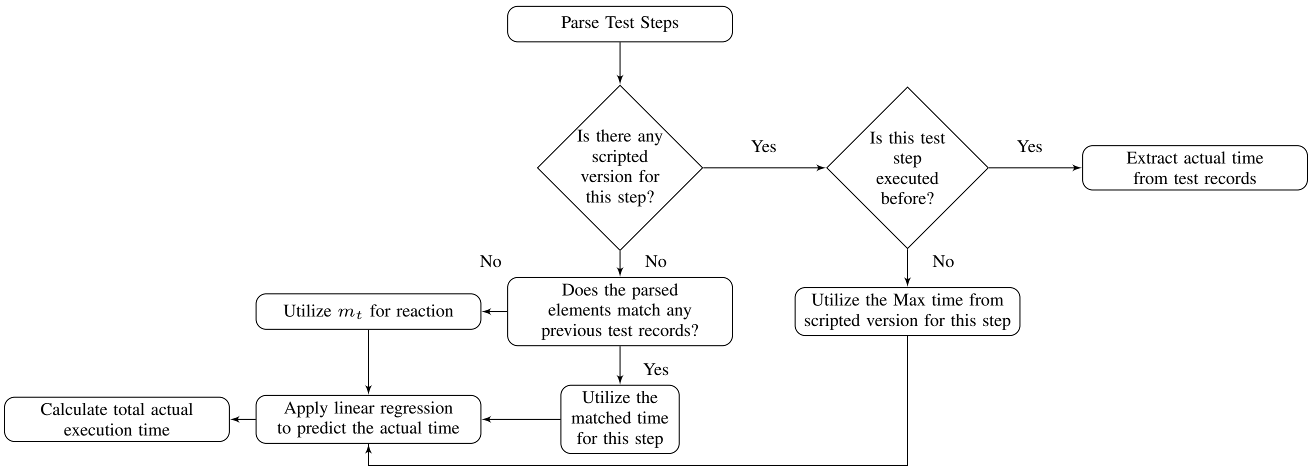
Parse Test Steps
Is there any scripted version for this step?
Yes
Is this test step executed before?
Yes
Extract actual time from test records
No
No
No
Does the parsed elements match any previous test records?
Utilize m_t for reaction
Utilize the Max time from scripted version for this step
Yes
Utilize the matched time for this step
Apply linear regression to predict the actual time
Calculate total actual execution time

Figure 1: The steps of the proposed approach

3.1. Estimation Algorithm

Since the specifications of manual test cases are written in English natural language, we need to apply some NLP (natural language processing) methods for analyzing test cases. As Table 1 represents, a step in a test case consists of several test activities. To identify and distinguish the activities, there are some keywords per test step that we need to first analyze. Therefore, we utilize NLTK Python platform for parsing various elements of each test step first. We classify *Verb* and *Objective Argument* as the most critical elements per test step distinguishing different activities.

The parsed elements are then saved in a database. Thus a unit of time will be assigned as MT for each parsed element. In the present work and use-case, as we observed, a subset of manual test cases have been scripted by the testers, therefore the value of MT is available for some of the parsed elements from their scripted version. In the next step, the *estimation* algorithm searches for matching between the parsed elements extracted from a test step and the set of elements from the steps of previously executed test cases to determine and assign the required execution time for test activities. In any case, if there exist no matched data for the parsed elements in the test step, we assign the baseline execution time m_t for those elements. As mentioned earlier, the value of m_t is the system's baseline response time and should be determined by analyzing the historical data, testing environment and system properties.

Furthermore, the value of MT and AT would be compared by the *estimation* algorithm continuously. As the last step, the *estimation* algorithm calculates the total waiting times between the steps and adds the values for the parsed elements to them. As the result, an MT value is determined for the each step of newly parsed test cases (and also AT, if a matching activity is executed before).

3.2. Prediction Algorithm

As explained before, the actual execution time (AT) is system dependent and shows the real time that a system under test takes to respond to an activity. Assume a set of test cases will be executed in system A. By applying the *estimation* algorithm on the test cases we are able to estimate the maximum execution time as described above. Via analyzing the historical data of previous executed test cases on system A and comparing them with the maximum execution time, we can predict the actual time that system A needs for execution. Through re-defining Equation 1 we calculate the total actual execution time (T'_n) for test case TC.

Let $t'_1, t'_2, \ldots, t'_n$ represent the actual execution time (also $t_1, t_2, \ldots, t_n$ represent the maximum execution time) for performing various test activities in test case TC, then:

$$T'_n = \sum_{j=1}^{n} t'_j \tag{2}$$

where $t'_1 \leq t_1, t'_2 \leq t_2, \ldots, t'_n \leq t_n$ and also $T'_n \leq T_n$. Suppose $(t_1, t'_1), (t_2, t'_2) \ldots (t_n, t'_n)$ represents MT and AT for test steps in test case TC which has been executed in system A, then:

$$t' = mt + h \tag{3}$$

where n is number of test steps, m is the slope of the linear regression line and h is the t'-intercept and can be calculated by utilizing Equations 4, 5 as follow:

$$m = \frac{n \sum_{i=1}^{n} t_i t'_i - (\sum_{i=1}^{n} t_i)(\sum_{i=1}^{n} t'_i)}{n \sum_{i=1}^{n} t_i^2 - (\sum_{i=1}^{n} t_i)^2} \tag{4}$$

$$h = \frac{(\sum_{i=1}^{n} t_i^2)(\sum_{i=1}^{n} t'_i) - (\sum_{i=1}^{n} t_i)(\sum_{i=1}^{n} t_i t'_i)}{n \sum_{i=1}^{n} t_i^2 - (\sum_{i=1}^{n} t_i)^2} \tag{5}$$

To fulfill the conditions in Equations 3, the test cases should have at least 3 steps with at least one distinct values of t.

In addition the percentage of predictions error (ε) can be calculated as:

$$\varepsilon = \frac{|Predicted\ time - Actual\ time|}{Actual\ time} \times 100 \tag{6}$$

4. Proof of Concept

This section contains an evaluation of the feasibility of our proposed method by applying it on one manual test case from the safety critical train control management system from our industrial partner. Table 2 presents the parsed elements in the analyzed test case, maximum execution time (MT) and also waiting time between steps (WT) which have been estimated by the *estimation* algorithm through searching in our database.

However, the *estimation* algorithm maps the parsed element in the test case with some similar test activities in other test cases. Moreover, for some activities which do not exist in our database, we set the baseline time value as $m_t = 3$ seconds, which was set in consultation with our industrial partner. The parsed elements (activities) are meant to be executed in a sequence per test step.

TABLE 2: Feasibility results of our approach.

TC Name	Exterior Light Function			
TC ID	TC-ExtLights-008			
Step	Verb	Objective Argument	MT	WT
1	Login	IDU, " maintainer no driving", A_1	4	0
2	Logout	removing key card, cab A_1	23	0
3	Login	IDU, driver, cab A_1	42	0
4	Login	IDU, driver, cab A_2	42	0
5	Logout	cab, removing key card, A_2	5	0
6	Login, wait	IDU driver, cab A_1	57	20
7	Press	full beam, button, driver, desk	9	0
8	Press	button, complete driving	13	0
9	Press	emergency, stop button	6	0
10	Restore	emergency, stop button, cab A_1	$m_t = 3$	0
11	Wait	23 seconds	23	23

Step number 10 in Table 2 is an activity which was not accessible in our database, which implies that this activity never executed before, therefore, the baseline time value $(m_t = 3)$ has been assigned for this step. However, step 11 consists of just 23 seconds waiting time. All other test activities in Table 2 have been matched by the *estimation* algorithm in our database, which implies that those activities have been executed before in other test cases. For instance, step 3 in Table 2 is equal to step 8 in another test case (*Route cycle* A_1), which has been executed earlier than this test case. As the second part of our approach, we are going to predict the actual execution time (AT) for this test case.

Since, AT is a system dependent time, we need to check some log files for the previous execution in the same system that the test case in Table 2 would be executed. By running the regression analysis method on MATLAB and through analyzing the log files, we predict the actual time that the test case would take.

Moreover, this test case has been executed two times on the same system. We also analyzed the log files for the test case after both executions. The result for the executions and also our prediction values has been summarized in Table 3. Step number 10 in Table 2 is an activity which was not accessible in our database, which implies that this activity never executed before, therefore, the baseline time value $(m_t = 3)$ has been assigned for this step. However, step 11 consists of just 23 seconds waiting time. All other test activities in Table 2 have been matched by the *estimation* algorithm in our database, which implies that those activities have been executed before in other test cases. For instance, step 3 in Table 2 is equal to step 8 in another test case (*Route cycle* A_1), which has been executed earlier than this test case. As the second part of our approach, we are going to predict the actual execution time (AT) for this test case. Since, AT is a system dependent time, we need to check some log files for the previous execution in the same system that the test case in Table 2 would be executed.

By running the regression analysis method on MATLAB and through analyzing the log files, we predict the actual time that the test case would take. Moreover, this test case has been executed two times on the same system. We also analyzed the log files for the test case after both executions. The result for the executions and also our prediction values has been summarized in Table 3.

TABLE 3: Actual and Predicted Execution Time

	Actual Time		
Step	Execution 1	Execution 2	Predicted Time
1	2	2	3.14
2	22	21	20.6
3	40	39	38
4	40	39	38
5	3	5	4.6
6	40	49	51.8
7	5	6	7.73
8	11	11	11.4
9	4	4	4.98
10	1	2	2.23
11	23	20	20.6
Total	191	198	203.8
ε	6.70 %	2.92 %	

As we can see in Table 3, the values for AT are different in execution 1 and 2 on the same system. Moreover the values of AT (in both executions) and also the predicted time are less than or equal to MT values in Table 2. Further, the percentage of prediction error (ε) has be calculated by applying Equation 6, which can be used to compare the accuracy of the predicted time in execution 1, 2. As can be observed, the predicted times calculated by our approach which have been confirmed as good enough, very useful by our industrial partner, and with low prediction error are even closer to the actual values in execution 2 (lower prediction error).

As we can see in Table 3, the values for AT are different in execution 1 and 2 on the same system. Moreover the values of AT (in both execution) and also the predicted time are less than or equal to MT values in Table 2. Further, the percentage of prediction error (ε) has be calculated by applying Equation 6, which can be used to compare the accuracy of the predicted time in execution 1, 2. As can be observed, the predicted times calculated by our approach which have been confirmed as good enough, very useful by our industrial partner, and with low prediction error, are even closer to the actual values in execution 2 (lower prediction error).

5. Conclusion and Future Work

In this work in progress paper, we introduced an approach for estimating and predicting execution time of test cases. We have analyzed the specification structure of manual test cases and also explained the operating steps and algorithms constituting the approach. The concept of maximum execution time (MT) has been proposed in our approach as an independent variable to help with predicting the required time for executing manual test cases using linear regression. Furthermore, we also introduced the percentage of predictions error (ε) as a means to measure the accuracy of the proposed approach.

As a future work, we investigate the use of this parameter in fine-tuning the approach and to further improve the closeness of the predictions to reality. In order to minimize the prediction error (ε), other interpolation methods such as polynomial and spline interpolation may also be applied in the *prediction* algorithm.

We need to consider that the required time for running a test case by testers manually depends on both the system characteristics and testers' skills. For instance, the required time that an inexperienced tester needs to perform an activity (e.g., finding a signal value) is more than an experienced tester. In this paper, we just focused on the required time that a system takes to perform test activities through analyzing test specifications to enable scheduling of manual test cases before execution.

Moreover, by utilizing MT, we are able to prioritize test case for execution. Since, the execution cost of test cases is a function of the execution time, we are also able to estimate the total required cost for performing test activities by using this study. We have already started evaluating our approach on a large set of test cases at our case organization (Bombardier Transportation).

6. Acknowledgements

This work was supported by ECSEL & VINNOVA through H2020 ECSEL Project MegaM@RT2 (No.737494) and the Swedish Knowledge Foundation (KKS) through the TOCSYC (20130085) and the TestMine (20160139) projects.

References

[1] P. Pocatilu, "Automated Software Testing Process," *Economy Informatics*, vol. 1, 2002.

[2] S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal, and D. Sundmark, *Cost-Benefit Analysis of Using Dependency Knowledge at Integration Testing*. Springer International Publishing, 2016, pp. 268–284.

[3] S. Tahvili, *A Decision Support System for Integration Test Selection*, October 2016, Licentiate Thesis Dissertation, Mälardalen University, Sweden.

[4] F. Pop, C. Dobre, and V. Cristea, "Genetic Algorithm for DAG Scheduling in Grid Environments," in *IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009.

[5] R. Torkar, N. Awan, A. Alvi, and W. Afzal, "Predicting Software Test Effort in Iterative Development using a Dynamic Bayesian Network," in *Proceedings of the 21st International Symposium on Software Reliability Engineering – Industry Track*, 2010.

[6] E. Aranha and P. Borba, "An Estimation Model for Test Execution Effort," in *1st International Symposium on Empirical Software Engineering and Measurement*, 2007.

[7] E. Nikolaropoulos, "Testing Safety-Critical Software," *Hewlett-Packard Journal*, vol. 3, p. 48, 1997.

[8] R. Craig and S. Jaskiel, *Systematic Software Testing*, ser. Artech House ITS library. Artech House, 2002.