Predicting software test effort in iterative development using a dynamic Bayesian network

R. Torkar, N. M. Awan, A. K. Alvi and W. Afzal
Blekinge Institute of Technology
S-371 79 Karlskrona, Sweden
Email: richard.torkar@bth.se

Abstract—Projects following iterative software development methodologies must still be managed in a way as to maximize quality and minimize costs. However, there are indications that predicting test effort in iterative development is challenging and currently there seem to be no models for test effort prediction.

This paper introduces and validates a dynamic Bayesian network for predicting test effort in iterative software development.

The proposed model is validated by the use of data from two industrial projects. The accuracy of the results has been verified through different prediction accuracy measurements and statistical tests.

The results from the validation confirm that the model has the ability to predict test effort in iterative projects accurately.

Keywords—effort prediction; estimations; testing; agile; dynamic Bayesian network

I. INTRODUCTION

Many times, software test teams are responsible to handle different test activities such as estimation of test effort, bug identification, test case design, test tool selection, test team selection, etc. Hence, test managers need to plan the schedule accurately and efficiently to utilize the testing resources in order to meet deadlines. An accurate and efficient test effort estimation method could help test managers in completing projects successfully.

Inaccurate effort estimates may lead to poor quality, customer dissatisfaction, and developers’ frustration. Project uncertainty, use of estimation development processes, use of estimation management processes and the estimator’s experience, are a few factors that can affect effort estimation errors [1]. In iterative development, as in traditional development, accurate project estimates of effort and duration are of high importance [2]. However, it is challenging to determine the amount of effort required to perform test and defect fixing activities. On the one hand, if testers spend too much effort in testing the schedule may be delayed, while, on the other hand, if they spend less effort on testing the quality can be affected [3]. In addition to this, incomplete artifacts also make it difficult to estimate effort and size [4, 5] and relatively little work has been done on modeling, planning and controlling iterative development methodologies [6], particularly in terms of planning and controlling the effort.

Some claim that since the iterations in iterative development cycles are frequent there would be no need for planning and estimating to the extent one can see in traditional development models. This argument, however, is not true; project plans and estimations provide a mechanism to keep track on project progress that helps to achieve project goals [7], and this is equally true for iterative development.

In this paper we present a model for project managers to use when predicting software test effort in iterative development. The model is simple in its nature, uses few variables and a straightforward nomenclature. By using a dynamic Bayesian network we are able to combine various data types and use these to better predict test effort in iterative projects. The tool has been validated on data collected from two projects in industry.

In the next section we cover related work, while Section III presents our model design. The model is then validated by a pilot test (Section IV) before conducting an industry validation (Section V). Finally, we conclude the paper in Section VI.

II. RELATED WORK

In recent years, several researchers [8–10] have used Bayesian network (BN) to model uncertainties in software projects, e.g. BNs have been used successfully to support the managerial decision-making [11], allowing the project manager to trade-off the resources used against the output in terms of functionality and quality [12]. What follows next is a summary of relevant work in this field.

In [8], Rees et al. used Bayesian (graphical) models to model the uncertainties involved in software testing, quality process and provide support to test managers to use the model.

Fenton and Neil [13], and Hearty et al. [14], have all shown that BNs have many benefits over classical or regression-based models. The Bayesian network approach does not rely on a single point value; instead of predicting a single value of the variable a BN provides a complete probability distribution.

In 2006, Wang et al. presented a project level estimation model framework using a Bayesian belief network (BBN)
Their BBN used four sub-models: component estimation; test effectiveness estimation; residual defect estimation; and test estimation. By using this framework, estimation of quality, effort, schedule and scope could be determined at both project level and in specific phases; hence, providing support for managerial decision-making. However, the problem with Wang’s et al. framework was that it had not been validated in any real project and, further, was not tailored to any particular development methodology. Moreover, no statistical and prediction accuracy measures were performed.

In 2009, Hearty et al. [14], presented a Bayesian network causal model for the extreme programming (XP) methodology. They showed how, without using any additional metrics collection programs, their model could learn from project data in order to make quantitative effort predictions and risk assessments. They validated their model in an industry project. However, the model was validated with one instance of project data and only two XP practices were introduced into the model. They focused on project level predictions rather than specific testing processes.

In [9], Wooff et al. presented an approach for estimating software testing using a Bayesian graphical model. They conducted a case study and tried to solve software testing problems with the help of slightly more formal mechanisms, such as logical structuring of software testing, test design and analysis process, incorporation and implication of test results, probabilistic and statistical treatment of uncertainties. The model used software testers’ expert opinion as main input.

Abrahamsson et al. [16], proposed an incremental, iteration-based effort prediction model for agile or iterative development methodology. They validated the model with two semi-industrial projects by conducting a case study. The results indicated that their model performed better compared to traditional approaches. However, their model was not used to estimate test effort, but rather it estimated development effort in general. Additionally, most of the participants in their case study were master students and new to the eXtreme Programming methodology and, hence, one could question the sample selection and the validity threats it might lead to.

As far as we know, no framework has been developed for test effort estimation in iterative development. In this paper we propose such a framework using a dynamic Bayesian network and in the next section we present the framework design.

III. FRAMEWORK DESIGN PROPOSAL

In this section, we explain the proposed framework design. However, first we introduce the reader to some key concepts, such as, Bayesian networks, dynamic Bayesian networks, notations, parameters and nodes.

A. Bayesian Networks

A Bayesian network is composed of nodes and directed arrows. Nodes represent variables while arrows (directed edges) represent casual relationship between the nodes. A node without a parent (i.e. root node) is defined by the prior probability distribution while nodes with parents (leaf nodes) are defined through conditional probability distribution (CPD). In other words, a Bayesian network is a high level representation of a joint probability distribution for variables used to build a model for a specific problem [17,18]. One advantage, with Bayesian networks, is the ability to combine sparse data, prior assumptions and expert judgment into one single model [14].

B. Dynamic Bayesian Networks

Dynamic Bayesian networks (DBN), on the other hand, are the temporal extensions of the Bayesian networks, it extends the Bayesian networks by adding temporal dimensions (time) to the model [14,19]. DBNs explicitly model change over time, while BNs do not have any explicit temporal relationships between nodes or variables.

A DBN consist of a sequence of identical BNs, \( Z_t \), where \( t = 1, 2, 3, \ldots, n \). Each \( Z_t \) represents a process that is modeled at time \( t \) (called a time slice) [14]. Time slices are connected by temporal links; if two time slice structures are identical, and the temporal links are the same, then the model is a repetitive temporal model. Furthermore, if conditional probabilities are also identical, then we call it a dynamic Bayesian network [20].

Next we look at the requirements we set on our model design.

C. Model Requirements

The following requirements must be met by the model:

- To minimize the complexity, the model must be small, simple and easy to replicate in all iterations.
- The model should be able to learn from project data and expert opinion.
- The model should be able to respond to any observation when entered into the model.
- The model should be able to learn in different scenarios such as when a project is failing (e.g. missing deadlines) or succeeding (e.g. meeting deadlines), as well as performing on an average.
- The model should be able to predict test effort by considering the impact of different factors such as test process effectiveness, test team, test tools, etc.
- The model should be able to handle different kind of data e.g. integer and rank values.

D. Model Limitations

The model has the following limitations:
The model covers only iterative software development projects (and as such has not been validated for other methodologies).

The model is unable to predict test effort in the first iteration.

The model requires $N$th iteration data to predict for $N + 1, N + 2, \ldots, N + n$ iterations.

Predictions will be valid for the same project whose data is incorporated into the model.

**E. Model Design**

Fenton and Pfleeger [21] has described the software measurements as divided into: processes (related to software activities such as development and support); products (involves the deliverables such as requirements, design and code); and resources (involves the assets such as people, tools and equipments).

According to [21], all predictive and estimation models fall within these three classes. In our case, we have used process and resource measurements. To minimize the model’s complexity, we have chosen not to use product measurements (that category of data could later be incorporated into the model in order to increase prediction accuracy).

The proposed Bayesian network model (Figure 1) is composed of two main sections i.e. test process overall effectiveness ($\varepsilon$) and test effort ($E$), where the test effort ($E$) node is controlled by iteration length ($l$), team size ($s$) and test process overall effectiveness ($\varepsilon$).

Test process overall effectiveness contains two nodes, i.e. test process rework effectiveness and test process effectiveness. Further, rework test effectiveness is controlled by two nodes such as rework test effort and rework test process quality nodes. In fragment 2 (see Figure 1), test process effectiveness is controlled by test process quality which is further controlled by test tool quality, test team experience and test case effectiveness nodes. Finally, test case effectiveness is controlled by the number of defects found by test cases and the total number of defects found by nodes. Figure 1 shows how different nodes are linked to each other in the model.

Previous existing research work has been used to build the proposed model, more detailed information can be found in [14,15,22]. Our proposed model is different from the other existing models as discussed in Section II. Rather than creating a complex and large model, we have constructed a simple and small model with few, and what we believe to be, important nodes or variables that would be easily accessible in industry.

**F. Model Notations**

We have used different notations and symbols to represent model nodes, such as $E$ for test effort and $\varepsilon$ for test process overall effectiveness (see Table I). Subscripts are used with node symbols to represent a specific iteration number. For example $E_3$ and $t_4$ represent test effort and team size for iterations number 3 and 4, respectively.

**IV. INITIAL VALIDATION**

After the completion of the model design, it is important to conduct an initial validation of the model by performing a pilot test. The purpose of the initial validation is to ensure that a minimum level of quality has been reached, in terms of prediction accuracy, so that one, then, can perform the next step, i.e. studies in industry (Section V).

**A. Model Behavior**

To test the model’s behavior, the authors designed a project $X$ consisting of a total of eight iterations with 120 hours of available effort for a single iteration. The model was solely based on initial settings, where no data or observations were entered into the model. The model was executed and the test effort ($E$) output mean, median and standard deviation (SD) values were observed in order to evaluate the model’s prediction behavior, which is also known as baseline scenario (Table II and Figure 2).

The key parameters of the model such as iteration length ($l$), test team size ($s$) and test process overall effectiveness ($\varepsilon$) obviously allowed the test effort graph to increase gradually with each iteration and, indeed, to model such kind of behavior was one of the main objectives with the pilot study.
A number of different scenarios were then generated by entering different observations into the test effort ($E$) parameter of the model. In this way, we evaluated the predicted and learned values of $E$’s future iterations (as discussed in the next section).

### B. Calibration of Parameters

To evaluate the model’s behavior, it was important to consider various model parameters under different scenarios [25]. Therefore, we created three different scenarios in the model named success, average, and failing. The success scenario represents that the project is making progress (better than according to plan), the average scenario represents that the project is making progress better than according to plan, while the failing scenario represents a project failing in some way.

Table I

<table>
<thead>
<tr>
<th>Name</th>
<th>Descr.</th>
<th>Type</th>
<th>Range/Value</th>
<th>Formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>$tpe_i$</td>
<td>rework test effort</td>
<td>ranked</td>
<td>enough, not enough, more than enough [23]</td>
<td>-</td>
</tr>
<tr>
<td>$rtq_i$</td>
<td>rework test process quality</td>
<td>“”</td>
<td>very low, low, average, high, very high [23]</td>
<td>-</td>
</tr>
<tr>
<td>$re_i$</td>
<td>rework test effectiveness</td>
<td>“”</td>
<td>“”</td>
<td>$re_i = \frac{tpe_i + rtq_i}{2}$</td>
</tr>
<tr>
<td>$tte_i$</td>
<td>test tool quality</td>
<td>“”</td>
<td>“”</td>
<td>-</td>
</tr>
<tr>
<td>$tqe_i$</td>
<td>test team experience</td>
<td>“”</td>
<td>“”</td>
<td>-</td>
</tr>
<tr>
<td>$tpq_i$</td>
<td>test process effectiveness</td>
<td>“”</td>
<td>“”</td>
<td>$tpq_i = \frac{wmean(0, tte_i, 0, tqe_i)}{2}$</td>
</tr>
<tr>
<td>$b_i$</td>
<td>#faults found by test cases</td>
<td>numeric</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$tb_i$</td>
<td>total # of faults</td>
<td>“”</td>
<td>&gt;= 0</td>
<td>-</td>
</tr>
<tr>
<td>$tce_i$</td>
<td>test case effectiveness</td>
<td>”</td>
<td>0–100 (%)</td>
<td>$tce_i = \frac{b_i}{tb_i} \times 100$ [24]</td>
</tr>
<tr>
<td>$e_i$</td>
<td>test process overall effectiveness</td>
<td>”</td>
<td>0–1</td>
<td>$e_i = \frac{b_i}{tce_i}$</td>
</tr>
<tr>
<td>$l_i$</td>
<td>iteration length</td>
<td>”</td>
<td>&gt;= 1</td>
<td>-</td>
</tr>
<tr>
<td>$s_i$</td>
<td>test team size</td>
<td>”</td>
<td>”</td>
<td>$E_i = l_i \times s_i \times e_i$</td>
</tr>
<tr>
<td>$E_i$</td>
<td>test effort</td>
<td>”</td>
<td>”</td>
<td>-</td>
</tr>
</tbody>
</table>

The baseline scenarios’ predicted median values for $I_1$–$I_8$ is in the range of 35.65 to 40.23 as shown in Table II. However, we have assumed different values for $E$ under success, average and failing scenarios for $I_1$–$I_8$, i.e. in the range of 42–50, 34–41 and 2–6, respectively. Table III presents the test effort ($E$) in each iteration for the success, average and failing scenarios.

To observe model parameter learning, all values from Table III were entered into the model, no values for $I_7$ and $I_8$ were entered, allowing the model to predict the test effort ($E$) for these iterations. (See Figure 3 for a graph of baseline, success, average and failing scenarios.)

From the graphs shown in Figure 3, one can see that the average scenario is very close to the baseline scenario and begins to stabilize by $I_5$. The failing scenario on the other hand is far away from the baseline scenario. The low values of test effort for the failing scenario could be a reason for this behavior to occur. While in the success scenario $E$ quickly learned from the observations and, as is evident, the
predicted values for $I_7$ and $I_8$ are very stable at $E_7 = 39.07$ and $E_8 = 39.51$.

In Figure 4, different scenarios for test process overall effectiveness nodes are shown, and one can see that the failing scenario shows no improvement until $I_5$, i.e. no effectiveness improvement. The success scenario, on the other hand, shows high effectiveness as compared to the average and failing scenarios and, additionally, its predicted values for $I_7$ and $I_8$ are very stable.

As we have seen, the test effort ($E$) and test process overall effectiveness ($e$) propagates as observations are entered and then update their predictions accordingly, hence providing a first validation of the model. Similarly, different scenarios for other nodes such as rework test effort effectiveness ($rte$), test process effectiveness ($tpe$), etc. can also be generated.

V. INDUSTRY VALIDATION

In this section, we provide details on the industrial validation of the model. The validation was, initially, performed by conducting a series of interviews for data collection purpose, where the respondents had participated in two projects (Projects A and B). The data collected during the interviews was then used to perform a validation of the results with an accompanying analysis.

A. Project A

This section describes how project A data was incorporated in the model to predict test effort for later iterations. The data gathered from respondent A regarding project A is shown in Table IV. The respondent has managed a SCRUM project the last three years and the data was compiled after completing several interviews. The company where the respondent works focuses on several domains, such as, mobile applications, multimedia, telecom networks, software tools and enterprise systems.

There were two scenarios developed in the model, i.e. actual and predicted. Initially, from Table IV, only iteration length ($l$) and test team size ($s$) values of all six iterations were entered into both scenarios. The remaining variables, as seen in Table IV (i.e. test tool quality, test team experience, bugs found by test case, total bugs found, rework test effort and rework test process quality values from $I_1$ to $I_6$) were entered only into the actual scenario.

After entering the data from Table IV into the actual and predicted scenarios, the model was executed and test effort ($E$) node median values were observed for actual and predicted scenarios as shown in Table V.

With the help of Table V, and without entering observations into the ‘predicted’ scenario, a graph for test effort ($E$) node was generated for both ‘actual’ and ‘predicted’ scenarios. Figure 5 clearly shows that for $I_1$ the predicted values are too low and for $I_2$ the predicted values are too high as compared to the ‘actual’ values and, further, the ‘actual’ graph seems to be a bit unstable. The frequent changes in test tool quality and test team experience should account for this behavior.

Next, we examined the model’s ability to learn from the ‘real’ project’s data in order to improve its predictions. Thus, we took only the predicted scenario and entered data from the first two complete iterations: $I_1$ and $I_2$ (see Table IV). As new data was entered, propagation took place, allowing the distributions of key parameter to be updated, thus, affecting the prediction of future iterations (Table VI and Figure 6).

From Figure 6, we can see that the ‘predicted’ graph

| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$
|------|------|------|------|------|------
| $s_1$ | 3    | 3    | 3    | 3    | 3    |
| $t_{te}$ | M   | M    | H    | H    | H    |
| $t_{tb}$ | L   | M    | H    | VH   | VH   |
| $r_{te}$ | 0   | 0    | 5    | 10   | 10   |
| $r_{tb}$ | 0   | 10   | 10   | 10   | 10   |
| $rt_{te}$ | E   | E    | E    | E    | >E   |
| $rt_{tb}$ | M   | H    | H    | VH   | VH   |

Table V

PROJECT A—ACTUAL VS. PREDICTED TEST EFFORT (WITHOUT OBSERVATIONS ENTERED).

| Actual $M$ | 14.41 | 12.79 | 10.42 | 10.42 | 8.79 | 9.81 |
| Predicted $M$ | 10.40 | 10.84 | 11.09 | 11.25 | 11.57 | 11.46 |

Figure 5. Project A—Actual vs. predicted test effort.
Figure 1. Project A—Actual vs. predicted test effort (with observations \( I_1 \) & \( I_2 \) entered).

Figure 6. Project A—Actual vs. predicted test effort (without observations entered for \( I_1 \) & \( I_2 \)).

Table VI

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( I_4 )</th>
<th>( I_5 )</th>
<th>( I_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.42</td>
<td>8.79</td>
</tr>
<tr>
<td>Predicted ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.42</td>
<td>8.79</td>
</tr>
</tbody>
</table>

The graphs in Figure 7 clearly show the change in the graph for the predicted test effort when \( I_3 \) data was entered. The graph is settled and closer to the ‘actual’ graph and the predictions for \( I_3 \)–\( I_6 \) also improve as compared to prior predictions (Figure 6).

Figure 7. Project A—Actual vs. predicted test effort (with observations \( I_1 \)–\( I_3 \) entered).

Table VII

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( I_4 )</th>
<th>( I_5 )</th>
<th>( I_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.42</td>
<td>8.79</td>
</tr>
<tr>
<td>Predicted ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.81</td>
<td>11.05</td>
</tr>
</tbody>
</table>

B. Project B

In this section, we present the results from validating the model using data from project B. Table VIII shows the actual data of project B as gathered from respondent B. Respondent B has the last ten years been working at a large telecom company, where he has been responsible for test activities in several projects using agile methodologies.

Similar to project A, two scenarios were created in the model for project B, i.e. ‘actual’ and ‘predicted’. Initially, only iteration length \( (l) \) and test team size \( (s) \) values were entered into both scenarios. However, all other remaining variables from iteration \( I_1 \) to \( I_6 \) were entered into the ‘actual’ scenario. These variables include test tool quality, test team experience, bugs found by test case, total bugs found, rework test effort and rework test process quality. Finally, the model was executed and test effort’s \( (E) \) probability distribution median values were observed for both scenarios (Table IX).

Figure 8 shows the graph of the two scenarios as generated with the help of data given in Table IX. The graph was generated without entering any additional observations into the model. In Figure 8, we can see that the predicted graph values for \( I_2, I_3, I_5 \) and \( I_6 \) are too high, except for \( I_4 \) as compared to the ‘actual’ graph.

To then validate the model with the data from project B, we only used the ‘predicted’ scenario and entered data from \( I_1 \) and \( I_2 \) into the model (Table VIII). As we entered new data into the model, propagation took place and parameter distributions were updated. These distributions obviously affected the prediction of all future iterations, as shown in Table X.

Using Table X, a graph for both scenarios was generated as shown in Figure 9. We can see that the ‘predicted’ scenario decreases its values and moves closer to the ‘actual’ scenario and, therefore, the prediction of \( I_3 \)–\( I_6 \) improves as

Table VIII

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( I_4 )</th>
<th>( I_5 )</th>
<th>( I_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual ( M )</td>
<td>17.64</td>
<td>29.75</td>
<td>29.75</td>
<td>14.5</td>
<td>14.64</td>
</tr>
<tr>
<td>Predicted ( M )</td>
<td>19.17</td>
<td>33.39</td>
<td>34.17</td>
<td>14.5</td>
<td>17.89</td>
</tr>
</tbody>
</table>

Table IX

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( I_4 )</th>
<th>( I_5 )</th>
<th>( I_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.81</td>
<td>11.05</td>
</tr>
<tr>
<td>Predicted ( M )</td>
<td>14.41</td>
<td>12.19</td>
<td>10.42</td>
<td>10.81</td>
<td>11.05</td>
</tr>
</tbody>
</table>
Figure 8. Project B—Actual vs. predicted test effort (no observations entered).

Figure 9. Project B—Actual vs. predicted test effort (with observations $I_1$ & $I_2$ entered).

C. Results Validation

This section describes the model’s accuracy results by performing prediction accuracy measures and statistical test. We have used statistical software package SPSS 18.0 to perform all statistical calculations.

In order to validate the results from executing the model, we have performed the following prediction accuracy measures as suggested by [26–28]:

- Magnitude of Relative Error (MRE).
- Mean Magnitude of Relative Error (MMRE).
- Estimation Magnitude of Relative Error (EMRE).
- Mean of Estimation Magnitude of Relative Error (MEMRE).

MRE is a normalized measure of discrepancy between actual and predicted values, it provides the basis for MMRE calculations, and it can be defined as: $\text{MRE} = \frac{|E_{ai} - E_{pi}|}{E_{pi}}$.

Where $E_{ai}$ represents actual and $E_{pi}$ represents the predicted effort for the $i$th iteration. In our particular case, we are interested in the deviation in relation to the predicted values not to the actual values. Kitchenham et al. [28] suggest the use of MEMRE, where absolute residuals ($|E_{ai} - E_{pi}|$) are divided by estimate ($E_{pi}$). MEMRE is based on EMRE and can be formulated as: $\text{EMRE} = \frac{|E_{ai} - E_{pi}|}{E_{pi}}$. The mean of EMRE can then be formulated as: $\text{MMRE} = \frac{1}{n} \sum_{i=1}^{n} \text{MRE}_i$.

With the help of above definitions and measures, we have calculated MRE, MMRE, EMRE and MEMRE for projects $A$ and $B$.

1) Prediction Accuracy Measurements: In order to check the prediction accuracy for project $A$ (Table VII and Figure 7) and $B$ (Table X and Figure 9) the MEMRE values were calculated. The results are shown in Tables XI and XII.

It is suggested in literature that MEMRE $\leq 0.25$ is an indicator of good prediction models [26,28]. Table XII shows that MEMRE $= 0.0446$ (i.e. much less than 0.25). Hence, we can claim that the predicted test effort ($E$) results for projects $A$ and $B$ seem to be fairly accurate. Nevertheless, we have also performed a statistical test to further validate the predicted test effort ($E$) results.

2) Normality Test: This section provides details on normality test performed on projects $A$ and $B$ datasets. In this paper we have used the Shapiro-Wilk normality tests, descriptive statistics, histograms and box plots to investigate if the data is normally distributed or not. All normality tests were performed on project $A$ and $B$ datasets separately.

In Table XIII, we can see that the significant value for project $A$ is $0.476$ which is greater than $0.01$ (level of significance). Therefore the actual data is not normalized. The same applies to project $B$ ($0.011$).

### Table X

**PROJECT B—ACTUAL VS. PREDICTED TEST EFFORT (WITH OBSERVATIONS ENTERED FOR $I_1$ & $I_2$).**

<table>
<thead>
<tr>
<th></th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
<th>$I_5$</th>
<th>$I_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>17.64</td>
<td>29.75</td>
<td>29.75</td>
<td>14.5</td>
<td>14.64</td>
<td>14.64</td>
</tr>
<tr>
<td>Predicted</td>
<td>17.64</td>
<td>29.75</td>
<td>30.78</td>
<td>14.5</td>
<td>16.6</td>
<td>16.56</td>
</tr>
</tbody>
</table>

### Table XI

**PREDICTION ACCURACY MEASURES FOR PROJECTS $A$.**

<table>
<thead>
<tr>
<th></th>
<th>$E_{ai}$</th>
<th>$E_{pi}$</th>
<th>MRE</th>
<th>EMRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>14.41</td>
<td>14.41</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_2$</td>
<td>12.19</td>
<td>12.19</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_3$</td>
<td>10.42</td>
<td>10.42</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_4$</td>
<td>10.42</td>
<td>10.81</td>
<td>0.037</td>
<td>0.036</td>
</tr>
<tr>
<td>$I_5$</td>
<td>8.79</td>
<td>11.05</td>
<td>0.257</td>
<td>0.205</td>
</tr>
<tr>
<td>$I_6$</td>
<td>9.81</td>
<td>11.21</td>
<td>0.143</td>
<td>0.125</td>
</tr>
</tbody>
</table>

**MMRE=0.073** **MEMRE=0.061**

### Table XII

**PREDICTION ACCURACY MEASURES FOR PROJECT $B$.**

<table>
<thead>
<tr>
<th></th>
<th>$E_{ai}$</th>
<th>$E_{pi}$</th>
<th>MRE</th>
<th>EMRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>17.64</td>
<td>17.64</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_2$</td>
<td>29.75</td>
<td>29.75</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_3$</td>
<td>29.75</td>
<td>30.78</td>
<td>0.036</td>
<td>0.035</td>
</tr>
<tr>
<td>$I_4$</td>
<td>14.5</td>
<td>14.5</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_5$</td>
<td>14.64</td>
<td>16.6</td>
<td>0.133</td>
<td>0.118</td>
</tr>
<tr>
<td>$I_6$</td>
<td>14.64</td>
<td>16.56</td>
<td>0.131</td>
<td>0.119</td>
</tr>
</tbody>
</table>

**MMRE=0.2996** **MEMRE=0.0446**
Further, the significance values for the projects predicted values (0.097 and 0.030 respectively) are also greater than 0.01, providing indications that the predicted data, for both projects, is non-normal. Thus, in general, project A and B datasets are not fully normalized. Histograms and box plots also indicate similar results (Figures 10 and 11).

3) Significance Testing: The main objective of hypothesis testing is to see if it is possible to reject a null hypothesis with high significance. We observed that project A’s data was not normally distributed and, hence, we opted for a non-parametric test; in this case the Wilcoxon signed ranks test.

We defined the null hypothesis to be the difference (in median) between actual and predicted not equaling 0. The alternate hypothesis, then, is the median of differences between actual and predicted equaling 0. The null and alternate hypothesis can also be formulated more formally as:

\[ H_0 : Median_{ai} - Median_{pi} \neq 0 \]
\[ H_1 : Median_{ai} - Median_{pi} = 0 \]

After applying the Wilcoxon signed ranks test, we computed \( p \)-values for projects A and B (asymptotic significance, 2-tailed) equaling to 0.109 in both cases. We have used 0.01 as level of significance, since the \( p \)-value (0.109) is greater than 0.01, therefore we can reject the null hypothesis.

Further, the results confirmed that there is no significant difference between the actual and predicted test effort \( (E) \) values (with \( p = 0.01 \)) and, thus, we can reject \( H_0 \) and accept the alternative hypothesis. Additionally, the statistical test also confirmed that there is no significant difference between the medians of actual and predicted graph values.

VI. Conclusions

It is important to manage iterative projects in a way to maximize quality and minimize cost. To achieve this, accurate project estimates are of high importance. However, it is challenging to predict the amount of effort required to perform test activities in an iterative development. Therefore, to overcome such a challenge we present here one alternative model.

This paper’s main contribution was to introduce and validate a dynamic Bayesian network for predicting test effort in iterative software development. Predicting test effort enables the test manager to plan and control the test activities that involve test team, test tools, etc. Thus, delayed test activities and schedule overrun can be avoided.

The results, from using the model, were validated in several ways:

First, the model was evaluated by considering parameter learning and initial validation. The results showed that the model’s parameters learn from prior iterations and the model is able to make predictions for coming iterations.

Second, the model was validated by incorporating data from two industry projects. To evaluate the model’s predictions, two scenarios were constructed (actual and predicted). In the case of project A, initially data from two iterations were entered into the model and test effort prediction results were observed. When additional iteration data was used, the model learned quickly and the prediction results were more accurate and improved considerably. In the case of project B, two iteration data sets were used. The results show that the model performed well, and predictions were more accurate compared to the ‘actual’ scenario for all future iterations.

Finally, the results of the industrial model validation were verified through prediction accuracy measures (MRE and MMRE) and statistical tests. The prediction accuracy measurements indicate that the prediction results for both projects (A and B) are very good indeed. Moreover, a statistical test was performed using Wilcoxon signed rank test. The results confirmed that there was no significant difference between the medians of ‘actual’ and ‘predicted’ values for the test effort. Thus, we recommend using the proposed model to predict test effort in iterative projects.

VII. Future Work

Specific agile practices can be introduced into the proposed model such as SCRUM, XP, TDD, etc. Model behavior and prediction accuracy can also be further evaluated by implementing the model in a currently ongoing project in the industry, i.e. performing a dynamic validation as suggested by [29]. In this way, managers can incorporate data from real
projects into the model and evaluate prediction results which could affect their possibility to manage a project efficiently.

Further, the results of the model can also be validated by more data from different kind of industrial projects. Moreover, similar kinds of models, for predicting effort, can also be constructed for each phase of the software development life cycle, e.g. requirements and design.

REFERENCES


