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)

[15]. 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 casual 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 Nth 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 (e) and test effort (E), where the test effort (E) node is controlled by iteration length (l), team size (s) and test process overall effectiveness (e).

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 e 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.

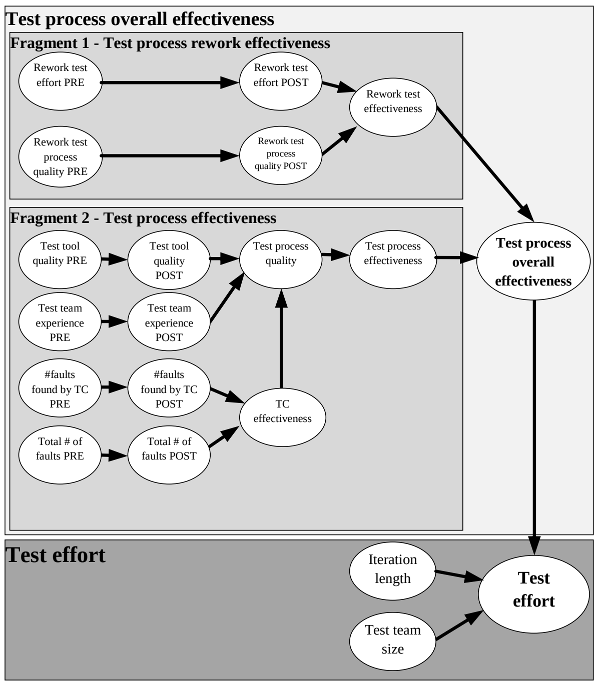
Test process overall effectiveness
Fragment 1 - Test process rework effectiveness
Rework test effort PRE
Rework test effort POST
Rework test effectiveness
Rework test process quality PRE
Rework test process quality POST
Fragment 2 - Test process effectiveness
Test tool quality PRE
Test tool quality POST
Test process quality
Test process effectiveness
Test process overall effectiveness
Test team experience PRE
Test team experience POST
#faults found by TC PRE
#faults found by TC POST
TC effectiveness
Total # of faults PRE
Total # of faults POST
Test effort
Iteration length
Test effort
Test team size

Figure 1. Proposed model design.

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 (e) 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.

Table I
VARIABLE NAME, DESCRIPTION, TYPES, RANGES/VALUES AND FORMULAS.

Name	Descr.	Type	Range/Value	Formulas
rte_i	rework test effort	ranked	enough, not enough, more than enough [23]	-
rtq_i	rework test process quality	”	very low, low, average, high, very high [23]	-
re_i	rework test effectiveness	”	”	$re_i = \frac{rte_i + rtq_i}{2}$
ttq_i	test tool quality	”	”	-
tte_i	test team experience	”	”	-
tpe_i	test process effectiveness	”	”	-
tpq_i	test process quality	”	”	tpq_i= $wmean(1, ttq_i, 5.0, tte_i, 1.0, tpe_i)$
b_i	#faults found by test cases	numeric	≥ 0	-
tb_i	total # of faults	”	”	-
tce_i	test case effectiveness	”	0–100 (%)	$tce_i = \frac{b_i}{tb_i} \times 100$ [24]
e_i	test process overall effectiveness	”	0–1	$e_i = \frac{re_i + tpe_i}{2}$
l_i	iteration length	”	≥ 1	-
s_i	test team size	”	”	-
E_i	test effort	”	”	$E_i = l_i \times s_i \times e_i$

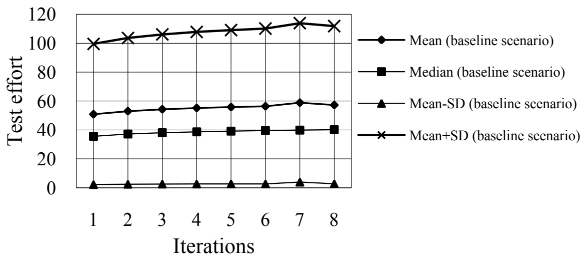

Figure 2. Test effort with baseline scenario.

Table II
BASELINE SCENARIO BASED ON INITIAL SETTINGS.

	Baseline				
Iterations	Mean	Median	SD	Mean − SD	Mean + SD
I_1	50.92	35.65	48.57	2.35	99.49
I_2	53.06	37.22	50.5	2.56	103.56
I_3	54.35	38.16	51.68	2.67	106.03
I_4	55.23	38.78	52.51	2.72	107.74
I_5	55.89	39.24	53.13	2.76	109.02
I_6	56.42	39.62	53.65	2.77	110.07
I_7	58.89	39.94	54.9	3.99	113.79
I_8	57.3	40.23	54.47	2.83	111.77

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 namned 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 going on as planned, while the failing scenario represents a project failing in some way.

Table III
TEST EFFORT IN EACH ITERATION $\{I_n\}$ FOR THE SUCCESS, AVERAGE AND FAILING SCENARIOS.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
Success	42	44	44	47	48	50	-	-
Average	34	35	35.5	36	40	41	-	-
Failing	2	2	3	4	4	6	-	-

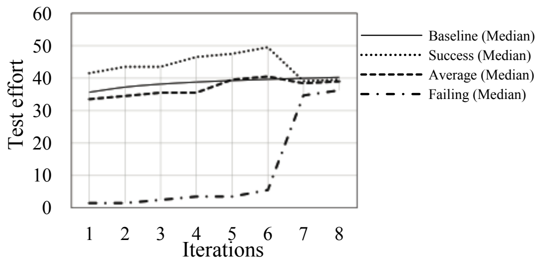

Figure 3. Test effort success, average, and failing scenarios.

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_6, 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

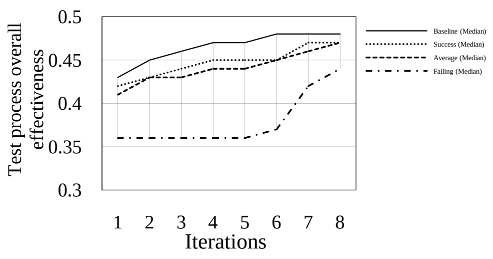

Figure 4. Test process overall effectiveness success, average, and failing scenarios.

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 they 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

Table IV
Project A dataset. (First column provides the variable names. L=Low, M=Medium, H=High, VH=Very High, E=Enough, >E=More than Enough.)

	I_1	I_2	I_3	I_4	I_5	I_6
l_i	10	10	10	10	10	10
s_i	3	3	3	3	3	3
ttq_i	M	M	H	H	H	H
tte_i	L	M	H	H	VH	VH
b_i	0	~ 5	~ 10	~ 10	~ 10	~ 10
tb_i	0	5	10	10	10	10
rte_i	E	E	E	E	E	>E
rtq_i	M	H	H	H	VH	VH

Table V
Project A—Actual vs. predicted test effort (without observations entered).

	I_1	I_2	I_3	I_4	I_5	I_6
Actual M	14.41	12.19	10.42	10.42	8.79	9.81
Predicted M	10.40	10.84	11.09	11.25	11.37	11.46

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_5 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

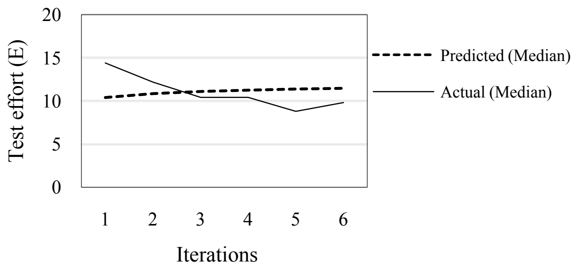

Figure 5. Project A—Actual vs. predicted test effort.

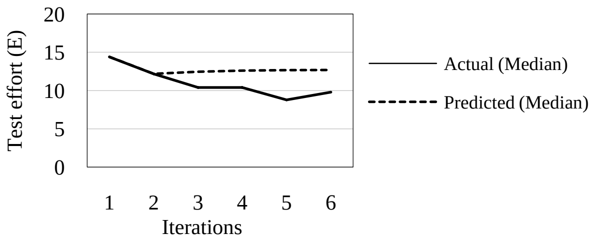

Figure 6. Project A—Actual vs. predicted test effort (with observations I_1 & I_2 entered).

Table VI
PROJECT A—ACTUAL VS. PREDICTED TEST EFFORT (WITH OBSERVATIONS ENTERED FOR I_1 & I_2).

	I_1	I_2	I_3	I_4	I_5	I_6
Actual M	14.41	12.19	10.42	10.42	8.79	9.81
Predicted M	14.41	12.19	12.47	12.61	12.67	12.7

has changed, when we entered data from I_1 and I_2 into the model. Further, the 'predicted' graph values are high as compared to the *a priori* predicted values given in Table V, instability of actual data and data provided by respondent A can be a reason for this. Next, I_3 data was entered into the model and the model was executed again. The parameter probabilities were updated and predictions improved as a result (Table VII and Figure 7).

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_4–I_6 also improve as compared to prior predictions (Figure 6).

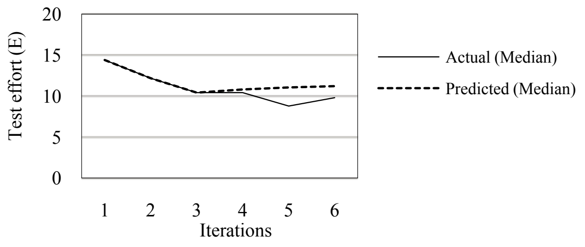

Figure 7. Project A—Actual vs. predicted test effort (with observations I_1–I_3 entered).

Table VII
PROJECT A—ACTUAL VS. PREDICTED TEST EFFORT (WITH OBSERVATIONS ENTERED FOR I_1–I_3).

	I_1	I_2	I_3	I_4	I_5	I_6
Actual M	14.41	12.19	10.42	10.42	8.79	9.81
Predicted M	14.41	12.19	10.42	10.81	11.05	11.21

Table VIII
PROJECT B DATASET. (FIRST COLUMN PROVIDES THE VARIABLE NAMES.)

	I_1	I_2	I_3	I_4	I_5	I_6
l_i	30	30	30	15	15	15
s_i	2	2.5	2.5	2.5	2.5	2.5
ttq_i	H	H	H	H	H	H
tte_i	VH	VH	VH	VH	VH	VH
b_i	0	0	0	2	2	2
tb_i	15	13	17	16	3	2
rte_i	E	E	E	E	E	E
rtq_i	M	M	M	M	M	M

Table IX
PROJECT B—ACTUAL VS. PREDICTED TEST EFFORT (WITHOUT OBSERVATIONS ENTERED).

	I_1	I_2	I_3	I_4	I_5	I_6
Actual M	17.64	29.75	29.75	14.5	14.64	14.64
Predicted M	19.17	33.39	34.17	14.5	17.89	17.84

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

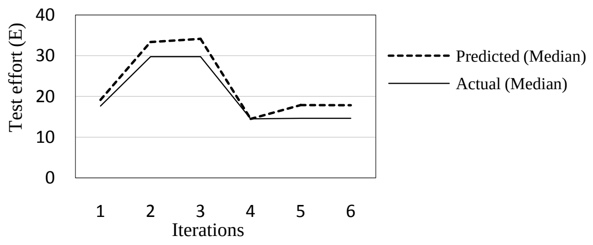

Figure 8. Project B—Actual vs. predicted test effort (no observations entered).

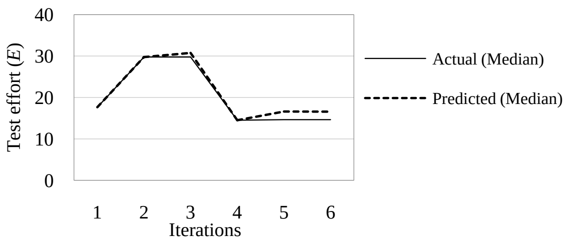

Figure 9. Project B—Actual vs. predicted test effort (with observations I_1 & I_2 entered).

a result.

The prediction results for project B are more stable as compared to the predictions of project A. The test effort (E), in Figures 8 and 9, shows improvement in the prediction scenario, but there is a need to validate the accuracy of the results as will be described in the next section.

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: MRE $= \frac{|E_{ai}-E_{pi}|}{E_{ai}}$.

Where E_{ai} represents actual and E_{pi} represents the predicted effort for the ith 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 it can be formulated as: EMRE $= \frac{|E_{ai}-E_{pi}|}{E_{pi}}$. The mean of EMRE can then be formulated as: MMRE $= \frac{1}{n}\sum_{i=1}^{i=n}$ 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 ≤ 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).

	I_1	I_2	I_3	I_4	I_5	I_6
Actual M	17.64	29.75	29.75	14.5	14.64	14.64
Predicted M	17.64	29.75	30.78	14.5	16.6	16.56

Table XI
PREDICTION ACCURACY MEASURES FOR PROJECTS A.

	E_{ai}	E_{pi}	MRE$_i$	EMRE$_i$
I_1	14.41	14.41	0.000	0.000
I_2	12.19	12.19	0.000	0.000
I_3	10.42	10.42	0.000	0.000
I_4	10.42	10.81	0.037	0.036
I_5	8.79	11.05	0.257	0.205
I_6	9.81	11.21	0.143	0.125
			MMRE=0.073	**MEMRE=0.061**

Table XII
PREDICTION ACCURACY MEASURES FOR PROJECT B.

	E_{ai}	E_{pi}	MRE$_i$	EMRE$_i$
I_1	17.64	17.64	0.0000	0.0000
I_2	29.75	29.75	0.0000	0.0000
I_3	29.75	30.78	0.0346	0.0335
I_4	14.5	14.5	0.0000	0.0000
I_5	14.64	16.6	0.1339	0.1181
I_6	14.64	16.56	0.1311	0.1159
			MMRE=0.2996	**MEMRE=0.0446**

Table XIII
SHAPIRO-WILK TEST FOR PROJECTS A AND B.

	Project A			Project B		
	Statistic	*df*	Sig.	Statistic	*df*	Sig.
Actual data	.916	6	**.476**	.723	6	**.011**
Predicted data	.825	6	**.097**	.768	6	**.030**

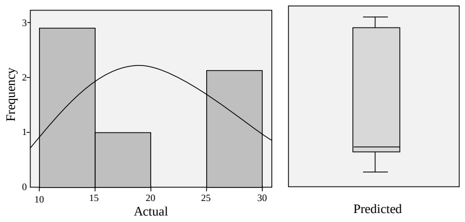

Figure 10. Project A—Histogram and boxplot of predictions.

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.

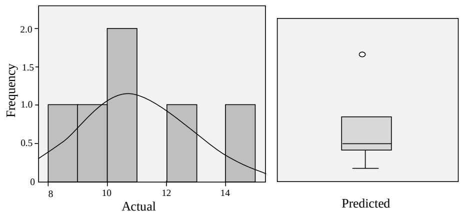

Figure 11. Project B—Histogram and boxplot of predictions.

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 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

[1] M. Jørgensen and D. I. K. Sjøberg, "The impact of customer expectation on software development effort estimates," *International Journal of Project Management*, vol. 22, no. 4, pp. 317–325, 2004.

[2] B. W. Boehm and R. E. Fairley, "Software estimation perspectives—Guest editors' introduction," *IEEE Software*, vol. 17, no. 6, 2000.

[3] L. Gou, Q. Wang, J. Yuan, Y. Yang, M. Li, and N. Jiang, "Quantitative defects management in iterative development with BiDefect," *Software Process: Improvement and Practice*, vol. 14, no. 4, pp. 227–241, 2009.

[4] M. Heričko and A. Živkovič, "The size and effort estimates in iterative development," *Information and Software Technology*, vol. 50, no. 7-8, pp. 772–781, 2008.

[5] S. Fricker, T. Gorschek, C. Byman, and A. Schmidle, "Handshaking with implementation proposals: Negotiating requirements understanding," *IEEE Software*, vol. 27, no. 2, pp. 72–80, 2010.

[6] O. Benediktsson and D. Dalcher, "Effort estimation in incremental software development," *IEE Proceedings - Software*, vol. 150, no. 6, pp. 351–358, 2003.

[7] K. Bittner and I. Spence, *Managing Iterative Software Development Projects*. Addison-Wesley Professional, 2006.

[8] K. Rees, F. Coolen, M. Goldstein, and D. Wooff, "Managing the uncertainties of software testing: A Bayesian approach," *Quality and Reliability Engineering International*, vol. 17, no. 3, pp. 191–203, 2001.

[9] D. A. Wooff, M. Goldstein, and F. P. A. Coolen, "Bayesian graphical models for software testing," *IEEE Transactions on Software Engineering*, vol. 28, no. 5, pp. 510–525, 2002.

[10] N. E. Fenton, P. Krause, and M. Neil, "Software measurement: Uncertainty and causal modeling," *IEEE Software*, vol. 19, no. 4, pp. 116–122, 2002.

[11] D. Settas, S. Bibi, P. Sfetsos, I. Stamelos, and V. Gerogiannis, "Using Bayesian belief networks to model software project management antipatterns," in *SERA '06: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 117–124.

[12] N. E. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M. Tailor, "Making resource decisions for software projects," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 397–406.

[13] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 675–689, 1999.

[14] P. Hearty, N. E. Fenton, D. Marquez, and M. Neil, "Predicting project velocity in XP using a learning dynamic Bayesian network model," *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 124–137, 2009.

[15] H. Wang, F. Peng, C. Zhang, and A. Pietschker, "Software project level estimation model framework based on Bayesian belief networks," in *QSIC '06: Proceedings of the Sixth International Conference on Quality Software*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 209–218.

[16] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi, "Effort prediction in iterative software development processes—Incremental versus global prediction models," in *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 344–353.

[17] A. Mittal, *Bayesian network technologies: Applications and graphical models*. Hershey, PA, USA: IGI Publishing, 2007.

[18] M. Valtorta and Y. Huang, "Identifiability in casual Bayesian networks: A gentle introduction," *Cybernetics and Systems*, vol. 39, no. 4, pp. 425–442, 2008.

[19] I. Flesch and P. Lucas, "Independence decomposition in dynamic Bayesian networks," in *ECSQARU '07: Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 560–571.

[20] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*. Springer Publishing Company, Inc., 2007.

[21] N. E. Fenton and S. L. Pfleeger, *Software metrics: A rigorous and practical approach*. Boston, MA, USA: International Thomson Computer Press, 1996.

[22] N. E. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra, "Predicting software defects in varying development lifecycles using Bayesian nets," *Information and Software Technology*, vol. 49, no. 1, pp. 32–43, 2007.

[23] N. E. Fenton, M. Neil, and J. G. Caballero, "Using ranked nodes to model qualitative judgments in Bayesian networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1420–1432, 2007.

[24] Y. Chernak, "Validating and improving test-case effectiveness," *IEEE Software*, vol. 18, no. 1, pp. 81–86, 2001.

[25] N. Lavesson and P. Davidsson, "Quantifying the impact of learning algorithm parameter tuning," in *21st AAAI National Conference on Artificial Intelligence*. Menlo Park, CA, USA: AAAI Press, 2006, pp. 395–400.

[26] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1986.

[27] C. van Koten and A. R. Gray, “Bayesian statistical effort prediction models for data-centred 4GL software development,” *Information and Software Technology*, vol. 48, no. 11, pp. 1056–1067, 2006.

[28] B. Kitchenham, L. Pickard, S. G. MacDonell, and M. J. Shepperd, “What accuracy statistics really measure,” *IEE Proceedings - Software*, vol. 148, no. 3, pp. 81–85, 2001.

[29] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, “A model for technology transfer in practice,” *IEEE Software*, vol. 23, no. 6, pp. 88–95, 2006.