Improved Bayesian Networks for Software Project Risk Assessment Using Dynamic Discretisation

Norman Fenton[1], Łukasz Radliński[2], Martin Neil[3]

[1,3] Queen Mary, University of London, UK
norman@dcs.qmul.ac.uk
[2] Queen Mary, University of London, UK
and Institute of Information Technology in Management, University of Szczecin, Poland
lukrad@dcs.qmul.ac.uk

Abstract. It is possible to build useful models for software project risk assessment based on Bayesian networks. A number of such models have been published and used and they provide valuable predictions for decision-makers. However, the accuracy of the published models is limited due to the fact that they are based on crudely discretised numeric nodes. In traditional Bayesian network tools such discretisation was inevitable; modelers had to decide in advance how to split a numeric range into appropriate intervals taking account of the trade-off between model efficiency and accuracy. However, recent a recent breakthrough algorithm now makes dynamic discretisation practical. We apply this algorithm to existing software project risk models. We compare the accuracy of predictions and calculation time for models with and without dynamic discretisation nodes.

1 Introduction

Between 2001 and 2004 the collaborative EC Project MODIST developed a software defect prediction model [4] using Bayesian Networks (BNs). A BN is a causal model normally displayed as a graph. The nodes of the graph represent uncertain variables and the arcs represent the causal/relevance relationships between the variables. There is a probability table for each node, specifying how the probability of each state of the variable depends on the states of its parents. The MODIST model (used by organisations such as Philips, QinetiQ and Israel Aircraft Industries) provided accurate predictions for the class of projects within the scope of the study. However, the extendibility of the model was constrained by a fundamental limitation of BN modelling technology, namely that every continuous variable had to be approximated by a set of discretised intervals (defined in advance). Since the MODIST project has been completed we have addressed the problem of modelling continuous nodes in BNs. A recent breakthrough algorithm (implemented in the AgenaRisk software toolset) now enables us to define continuous nodes without any restrictions on discretisation. The necessary discretisation is hidden from the user and calculated dynamically with great accuracy. In this paper we describe our work to rebuild the de-

fect prediction model using this approach to dynamic discretisation. In Section 2 we provide an overview of the MODIST model and explain the limitations due to static discretisation. In Section 3 we provide an overview of the dynamic discretisation approach and then apply it to construct a revised MODIST model in Section 4. We present a comparison of the results in Section 5.

2 Existing models for software project risk assessment

The defect prediction model developed in MODIST is shown in schematic form in Figure 1.

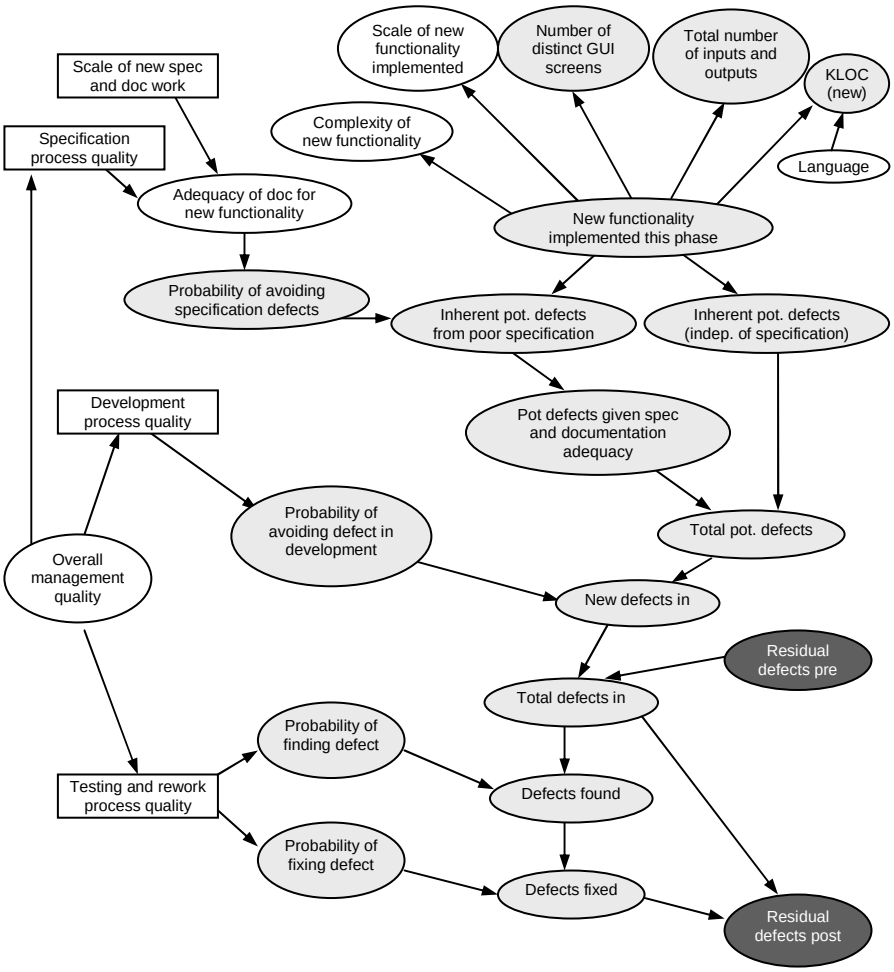

Fig. 1. Schematic view of defect prediction model; adopted from [2, 4]

Its main objective is prediction of various types of defects inserted or removed during various software development activities. All ellipses on this figure indicate a node of a Bayesian Net. Rectangles indicate subnets containing several additional nodes, which do not need to be shown here (since they are not important in this context and would cause unnecessary complexity).

This model can be used to predict defects in either of the following software development scenarios:

1. adding new functionality to existing code and/or documentation
2. creating new software from scratch (when no previous code and/or documentation exists).

The model in Figure 1 represents a single phase of software development that is made up of one or more of the following activities:

- specification/documentation,
- development (coding),
- testing and rework.

Such single phase models can be linked together to form a chain of phases which indicate major increments (milestone) in the whole project. This is the reason why in MODIST this model is called "phase-based defect prediction model". In this way we can model any software development lifecycle. More on modelling various lifecycles can be found in [3].

In common with many BN models this model contains a mixture of nodes that are qualitative (and are measured on a ranked scale) such as "Overall management quality" and nodes that are numeric, such as defects found and KLOC. Because generally BNs require numeric nodes to be discretised even if they represent continuous variables there is an inevitable problem of inaccuracy because a set of fixed intervals has to be defined in advance. To improve accuracy in predictions we have to split the whole range of possible values for a particular node into a larger number of intervals. The more intervals we have, the longer the calculation time – (since this includes generating the node probability table (NPT) from an expression in many cases). It is not simply a question of getting the right 'trade-off' because in many cases we need to assume an infinite scale for which, of course, there can never be a satisfactory discretisation.

One proposed solution to the problem has been to minimize the number of intervals by more heavily discretising in areas of expected higher probability, using wider intervals in other cases. This approach fails in a situation when we do not know in advance which values are more likely to occur. Such a situation is inevitable if we seek to use the models for projects beyond their original scope.

Table 1 illustrates node states in the MODIST model for two nodes describing size of the new software: "New Functionality" and "KLOC". Notice that there are several intervals where the ending value is around 50% or more higher than the starting value. The model cannot differentiate if we enter as an observation a starting, ending value or any other value between them. They are all treated as the same observation – middle of the interval.

There were two main reasons for defining such node states:

1. availability of empirical data that the model was later validated against
2. calculation time which was acceptable for the number of states.

The node “KLOC” contains intervals with high differences between starting and ending values. But those high differences are for values below 15 KLOC and over 200 KLOC (it was assumed that the KLOC in a single phase would never outside these boundaries). Hence, we can expect that predictions for software size between 15 and 200 KLOC will be more accurate than outside this range.

Table 1. Node for “New Functionality“ and “KLOC“

New Functionality				**KLOC (new)**			
Start	End	Interval Size	Percentage Difference Between Starting and Ending Values	Start	End	Interval Size	Percentage Difference Between Starting and Ending Values
0	24	25	-	0	0,5	0,5	-
25	49	25	**100,0%**	0,5	1	0,5	**100,0%**
50	74	25	**50,0%**	1	2	1	**100,0%**
75	99	25	33,3%	2	5	3	**150,0%**
100	124	25	25,0%	5	10	5	**100,0%**
125	149	25	20,0%	10	15	5	**50,0%**
150	199	50	33,3%	15	20	5	33,3%
200	298	99	**49,5%**	20	25	5	25,0%
299	399	101	33,8%	25	30	5	20,0%
400	499	100	25,0%	30	40	10	33,3%
500	749	250	**50,0%**	40	50	10	25,0%
750	999	250	33,3%	50	60	10	20,0%
1000	1499	500	**50,0%**	60	80	20	33,3%
1500	1999	500	33,3%	80	100	20	25,0%
2000	2999	1000	**50,0%**	100	125	25	25,0%
3000	4999	2000	**66,7%**	125	150	25	20,0%
5000	7999	3000	**60,0%**	150	175	25	16,7%
8000	12000	4001	**50,0%**	175	200	25	14,3%
12001	15999	3999	33,3%	200	300	100	**50,0%**
16000	19999	4000	25,0%	300	500	200	**66,7%**
20000	30000	10001	**50,0%**	500	10000	9500	**1900,0%**

For the “new functionality” node we cannot find any range of intervals with relatively low differences between lower and upper bound in an interval. This means that we will have relatively inaccurate predictions for most software size expressed in function points.

The defect prediction model contains several variables for predicting different types of defects. Most of them have similar states in terms both the number of states and their ranges. Table 2 illustrates intervals for one of them: “defects found”.

Table 2. Node states for “defects found“

Defects found				**Defects found (cont.)**			
Start	End	Interval Size	Percentage Difference Between Starting and Ending Values	Start	End	Interval Size	Percentage Difference Between Starting and Ending Values
1	4	4	**400,0%**	1500	2000	501	33,4%
5	19	15	**300,0%**	2001	3000	1000	**50,0%**
20	39	20	**100,0%**	3001	4000	1000	33,3%
40	59	20	**50,0%**	4001	5000	1000	25,0%
60	79	20	33,3%	5001	6000	1000	20,0%
80	99	20	25,0%	6001	7000	1000	16,7%
100	124	25	25,0%	7001	8000	1000	14,3%
125	149	25	20,0%	8001	9000	1000	12,5%
150	174	25	16,7%	9001	10000	1000	11,1%
175	199	25	14,3%	10001	11000	1000	10,0%
200	249	50	25,0%	11001	12000	1000	9,1%
250	299	50	20,0%	12001	13000	1000	8,3%
300	349	50	16,7%	13001	14000	1000	7,7%
350	399	50	14,3%	14001	15000	1000	7,1%
400	449	50	12,5%	15001	16000	1000	6,7%
450	499	50	11,1%	16001	17000	1000	6,2%
500	749	250	**50,0%**	17001	18000	1000	5,9%
750	999	250	33,3%	18001	19000	1000	5,6%
1000	1499	500	**50,0%**	19001	20000	1000	5,3%

3 Dynamic discretisation algorithm

The dynamic discretisation algorithm [5, 7] was developed as a way to solve the problems discussed in the previous section. The general outline of it is as follows:

1. Calculate the current marginal probability distribution for a node given its current discretisation.
2. Split that discrete state with the highest entropy error into two equally sized states.
3. Repeat steps 1 and 2 until converged or error level is acceptable.
4. Repeat steps 1, 2 and 3 for all nodes in the BN.

The algorithm has now been implemented in the AgenaRisk toolset [1]. Using this toolset we can simply set a numeric node as a simulation node without having to worry about defining intervals (it is sufficient to define a single interval [x, y] for any variable that is bounded below by x and above by y, while for infinite bounds we only need introduce one extra interval).

In the AgenaRisk tool we can specify the following simulation parameters:

- maximum number of iterations – this value defines how many iterations will be performed at maximum during calculation; it directly influences the number of intervals that will be created by the algorithm and thus calculation time,
- simulation convergence – the difference between the entropy error value between subsequent iterations; the lower convergence we set, the more accurate results we will have at the cost of computation time,
- sample size for ranked nodes – the higher value here reduces probabilities in tails for ranked node distributions at the cost of longer NPT generation process [1].

"Simulation convergence" can be set both as global parameter for all simulation nodes in the model or individually for selected nodes. In the second case the value of the parameter for a selected node overrides the global value for the whole model. If it is not set for individual nodes the global value is taken for calculation.

Currently there is no possibility to set the "maximum number of iterations" for a particular node. All nodes in a model use the global setting. This causes the same number of ranges to be generated by the dynamic discretisation algorithm for all simulation nodes in most of the cases. We cannot expect more intervals generated for selected nodes resulting in more accurate prediction there.

4 Revised software project risk models

Table 3 illustrates differences between node types for numeric nodes in the original and revised models.

We do not present number of states for numeric nodes in the revised model because they are not fixed. They rather depend on simulation parameters which are set by users.

In our model all numeric nodes are bound (do not have negative or positive infinity), so we set a single interval for those nodes.

Table 3. Numeric node types in original and revised models

Node	Original model			Revised model	
	Type of Interval	Simulation	Number of states	Type of Interval	Simulation
Prob avoiding spec defects	Continuous	No	7	Continuous	Yes
KLOC (new)	Continuous	No	21	Continuous	Yes
Total number of inputs and outputs	Integer	No	5	Integer	Yes
Number of distinct GUI screens	Integer	No	5	Integer	Yes
New functionality implemented this phase	Integer	No	21	Continuous	Yes
Inherent potential defects from poor spec	Integer	No	25	Integer	Yes
Inherent pot defects (indep. of spec)	Integer	No	25	Integer	Yes
Pot defects given spec and documentation adequacy	Integer	No	26	Integer	Yes
Total pot defects	Integer	No	26	Integer	Yes
New defects in	Integer	No	24	Integer	Yes
Total defects in	Integer	No	38	Integer	Yes
Defects found	Integer	No	38	Integer	Yes
Defects fixed	Integer	No	39	Integer	Yes
Residual defects pre	Integer	No	38	Integer	Yes
Residual defects post	Integer	No	38	Integer	Yes
Prob of avoiding defect in dev	Continuous	No	5	Continuous	Yes
Prob of finding defect	Continuous	No	5	Continuous	Yes
Prob of fixing defect	Continuous	No	5	Continuous	Yes

5 Comparison of results

All calculations have been performed on a computer with Pentium M 1.8 GHz Processor and 1 GB RAM under MS Windows XP Professional using AgenaRisk ver. 4.0.4. We ran calculations for the revised model using two values of parameter "maximum number of iterations": 10 and 25. We compared achieved results with the results achieved with the original model.

We observed very significant changes in predicted values for the revised and original model. Those differences varied among nodes and scenarios. Most of the predicted means and medians were significantly lower in the revised model than in the original (the range of those differences was from -3% to -80%). This result fixed a consistent bias that we found empirically when we ran the models outside the scope

of the MODIST project. Specifically, what was happening was that previously, outside the original scope, we were finding some probability mass in the end intervals. For example, an end interval like [10,000-infinity] might have a small probability mass, which without dynamic discretisation, will bias the central tendency statistics like the mean upwards. Only in a few cases did we observe an increase in predicted values. In all of them the differences were small – the highest was around 40%, but most of them did not reach 10%.

We could also observe a decrease in standard deviation for predicted distributions (from -8% to -80%). Partly this is explained by the model no longer suffering from the 'end interval' problem that also skewed the measures of central tendancy However, another reason is that dynamic discretisation fixes the problem whereby nodes that are defined by simple arithmetic functions had unnecessary variability introduced. For example, nodes like 'total potential defects', 'total defect in', 'residual defects post' no longer suffer from inaccuracies due entirely to discretisation errors affecting addition/subtraction.

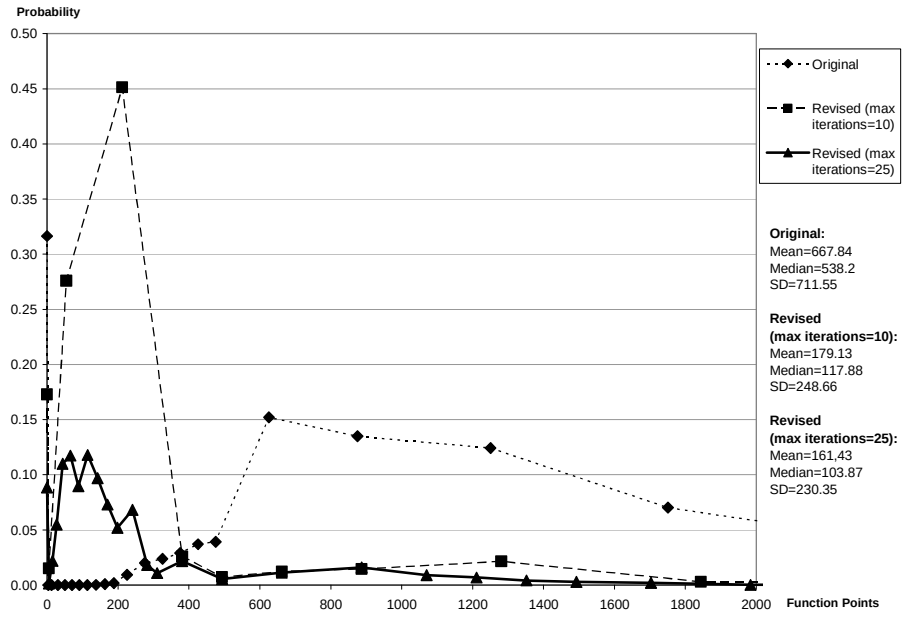

Fig. 2. Comparison of probability distributions for "residual defects post" for original and revised models for selected scenario

The dynamic discretisation algorithm creates node states in such way as to have narrow intervals within the area of highest probabilities and wide intervals where the probabilities are low (Fig. 2). This ensures greater accuracy for predicted values.

The number of intervals created for simulation nodes depends mainly on the parameter "maximum number of iterations". Figure 2 illustrates this. We can observe that in the areas of higher probability more intervals have been created.

Node states are fixed for the nodes not marked as simulation nodes. They do not change according to predicted values for those nodes.

We can observe that predicted values for the node "residual defects post" decreased significantly using the model with simulation nodes compared to the original. This occurred for both tested values of "maximum number of iterations". Predicted values for this node in both cases in the revised model were very similar (Fig. 2). Our results show this was also true in other scenarios and for other nodes.

Table 4. Comparison of calculation times for selected scenarios in original and revised model

Model	Time (in minutes)		Percentage difference in calculation times (compared to original model)	
	Average	Shortest	Average	Shortest
Original	0:13.1	0:11.1	-	-
Revised (Maximum number of iterations = 10)	0:18.7	0:15.7	42.8%	41.4%
Revised (Maximum number of iterations = 25)	2:03.1	1:34.8	839.0%	754.5%

We can observe the great difference between different settings of "maximum number of iterations" in calculation times (Table 4). When we compared calculation times for the revised model setting "maximum number of iterations" to 10 with the original model, we could observe that they increased by just over 40%. Although it was a significant increase in many cases it would make no real difference for end user.

However, calculation times increased very significantly when we set this parameter to 25 – around 8 times longer than in the original model. In this case we get only slightly more accurate predictions, so we must decide if much longer calculations can be compensated by only slightly higher precision.

The latest version of AgenaRisk (which we received just before finishing this research) contains optimizations to the algorithm which result in the times presented in Table 4 being generally halved. However, we cannot present precise information about as we were unable to perform extensive testing of the new algorithm.

6 Summary and future work

Results of our research have led us to the following conclusions:

1. Providing that we set a suitable value for the parameter "maximum number of iterations" the dynamic discretisation algorithm ensures greater accuracy of predicted values for simulation nodes than for nodes with fixed states.
2. Changing numeric node types to simulation nodes caused significant decrease in predicted "number of defects" and standard deviation (in several nodes). This re-

sult fixed a consistent (pessimistic) bias we had found empirically in projects outside the scope of MODIST.

3. Applying the dynamic discretisation algorithm does not force model builders to define node states at the time of creation of the model. This is a very useful feature especially in those cases when we do not know in advance in which ranges we should expect higher probabilities.
4. We can mix simulation and traditional nodes in a single model. We can define fixed node states for some of the nodes while setting others as simulation.
5. The cost of increased accuracy and model building simplicity that comes with dynamic discretisation is increased calculation timebut these increases are insignificant for values which still provide significant increases in accuracy..

Applying dynamic discretisation to the defect prediction model was one of a number of improvements we plan for the MODIST models. The next step will be to build an integrated model from the existing two developed in the MODIST project:

- defect prediction model,
- project level model (that contains, for example, resource information)

We also plan to apply dynamic discretisation to this integrated model and to extend it by incorporating other factors influencing the software development process.

References

1. Agena, AgenaRisk User Manual, 2005
2. Agena, Software Project Risk Models Manual, Ver. 01.00, 2004
3. Fenton N., Neil M., Marsh W., Hearty P., Krause P., Mishra R. Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets, to appear Information and Software Technology, 2006
4. MODIST BN models, http://www.modist.org.uk/docs/modist_bn_models.pdf
5. Neil M., Tailor M., Marquez D., Bayesian statistical inference using dynamic discretisation, RADAR Technical Report, 2005
6. Neil M., Tailor M., Marquez D., Fenton N., Hearty P., Modelling Dependable Systems using Hybrid Bayesian Networks, Proc. of First International Conference on Availability, Reliability and Security (ARES 2006), 20-22 April 2006, Vienna, Austria
7. Neil M., Tailor M., Marquez D., Inference in Hybrid Bayesian Networks using dynamic discretisation, RADAR Technical Report, 2005