Making Resource Decisions for Software Projects

Norman Fenton, William Marsh, Martin Neil, Patrick Cates, Simon Forey, and Manesh Tailor
Department of Computer Science, Queen Mary, University of London and Agena Ltd
{norman, william, martin, patrick, sf, manesh} @dcs.qmul.ac.uk

Abstract

Software metrics should support managerial decision making in software projects. We explain how traditional metrics approaches, such as regression-based models for cost estimation fall short of this goal. Instead, we describe a causal model (using a Bayesian network) which incorporates empirical data, but allows it to be interpreted and supplemented using expert judgement. We show how this causal model is used in a practical decision-support tool, allowing a project manager to trade-off the resources used against the outputs (delivered functionality, quality achieved) in a software project. The model and toolset have evolved in a number of collaborative projects and hence capture significant commercial input. Extensive validation trials are taking place among partners on the EC funded project MODIST (this includes Philips, Israel Aircraft Industries and QinetiQ) and the feedback so far has been very good. The estimates are sensible and the causal modelling approach enables decision-makers to reason in a way that is not possible with other project management and resource estimation tools. To ensure wide dissemination and validation a version of the toolset with the full underlying model is being made available for free to researchers.

1. Introduction

Among the many claimed benefits of software metrics, the most significant is that they are supposed to provide information to support managerial decision-making during the software lifecycle. Central to this decision-making is the trade-off between cost, quality (including functionality implemented) and project duration. The relative priority given to each of these attributes varies from project to project. On one project, the manager can chose to increase quality despite the cost; another project must achieve what is possible within a fixed budget. Effective decision-support to a project manager must support analysis of this trade-off.

In Section 2 we summarise existing approaches and explain why they are not sufficient for effective decision-support. In Section 3 we discuss the need for a causal approach to software prediction and introduce the notion of Bayesian nets, which we have used successfully in related commercial applications. In Sections 4 we provide a causal model (using Bayesian nets) for resource prediction and describe how it has been developed. In Section 5, we compare the model with published data and the predictions of popular resource-estimation models, showing the advantage of incorporating existing data into a causal model. A decision-support toolset allowing an end-user to interact with the model and to tailor it is described in Section 6.

2. The classic approach to resource estimation

The early resource prediction models (such as those of Putnam [29] and Boehm [5]) used size as the key variable for determining the effort required for a software development project. At first, Lines of Code (LOC or KLOC for thousands of lines of code) or related type metrics were used to measure size. However, the obvious drawbacks of this led to an explosion of interest in measures of software size (such as function points pioneered by Albrecht [1] and later investigated by Symons [31]) which were intended to be independent of programming language and to characterise the size of the problem rather than the solution. Other factors recognised as influencing the effort needed, including process and people attributes, are treated as cost drivers, adjusting the relationship between size and effort.

Despite these advances the overall approach to resource prediction has remained fundamentally unchanged since the early 1980's and has not been adopted widely. To provide better decision-support for managers we need to provide the following kinds of predictions:

- For a problem of this size, and given these limited resources, how likely am I to achieve a product of suitable quality?
- The model predicts that I need 4 people over 2 years to build a system of this kind of size. But I only have funding for 3 people over one year. If I cannot sacrifice quality, how good do the staff have to be to build the systems with the limited resources?

In the remainder of this paper we show how causal models, using Bayesian nets, can provide relevant predictions. Bayesian nets enable us to explicitly handle and measure the inevitable uncertainty that is pervasive in software engineering. Bayesian nets also enable us to obtain predictions with incomplete information, and hence provide a solution in cases where there is scarce data.

3. Causal Modelling with Bayesian Nets

A Bayesian net (BN) is a graph (such as that shown in Figure 1) together with an associated set of probability tables. The nodes represent uncertain variables and the arcs represent the causal/relevance relationships between the variables. There is a probability table for each node, providing the probabilities of each state of the variable. For variables without parents the table just contains the marginal probabilities while for variables with parents it has conditional probabilities for each combination of parent states.

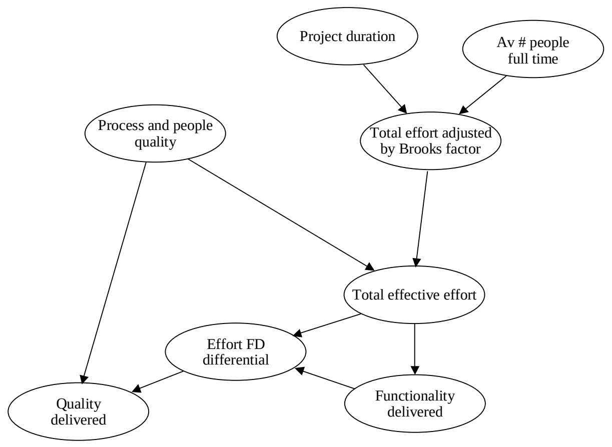


Figure 1. Project Resource BN (Simplified)

Although the underlying theory (Bayesian probability) has been around for a long time, executing realistic BN models was only first made possible in the late 1980s as a result of breakthrough algorithms and software tools that implement them [19]. Methods for *building* large-scale BNs is even more recent ([4, 28, 25]) but it is only such work that has made it possible to apply BNs to the problems of software engineering.

Drawing on this work in various commercial projects with Agena, Fenton and Neil have built BN-based applications that have proved the technology is both viable and effective. Several of these applications have been related to systems or software assessment. Especially significant was the TRACS tool [25] to assess vehicle reliability for QinetiQ (on behalf of the MOD) and the AID tool [7, 14] to predict software defects in consumer electronic products for Philips. Much of the modelling work described here has been done as part of the MODIST project [30], which extends the kind of ideas in AID. The toolset implementation has been based on Agena's Minerva technology that was extended to incorporate recent further research developments in building large-scale BNs that was undertaken in the SCULLY, SIMP and SCORE projects [15].

The BN in Figure 1 is a simplified version of the net we have developed for software project management. An early prototype of this BN was described in [8]. Like all BNs it was built using a mixture of data and expert judgements. Understanding cause and effect is a basic form of human knowledge, underlying the actions we take. For example, a project manager knows that increasing the number of people in the team *may* (there is some uncertainty) increase the delivered functionality. It is obvious that the relationship is not the other way round. The expert's understanding of cause and effect is used to connect the variables of the net with arcs drawn from cause to effect. Many of the relationships between variables have been prompted by empirical results described in a range of sources. These include [1, 2, 5, 12, 13, 17, 18, 20, 21, 22, 23, 24, 32].

To ensure that our model is consistent with these empirical findings, the probability tables in the net are constructed using data, whenever it is available. However, when there is missing data, or the data does not take account of all the causal influences, expert judgement must be used as well. In the next section, we explain how a model consistent with the empirical data is built. Later sections of the paper show we get improved decision support when we incorporate the empirical research results into a causal model.

4. A causal model for resource estimation

A causal model for resource estimation, in the form of a BN, has been constructed as part of the collaborative research project MODIST [30], involving five systems and software based organisations. Project management experts from these organisations have helped construct the model.

The full net extends the simplified net shown in Figure 1 in two ways. Firstly, the scope of the model has been expanded to address the concerns of organisations participating in the MODIST project. For example, MODIST specifically addresses risk management for 'distributed' software development (i.e. development at multiple locations in a large organisation) so the model includes the management of communications in such projects.

The complete model is too complex to show as a single BN (the full model and tool to execute and analyse it is available for download to researchers [16]). Its overall scope is shown in Figure 2, which shows six sub-nets. Each subnet contains variables relating to a particular aspect of the overall model.

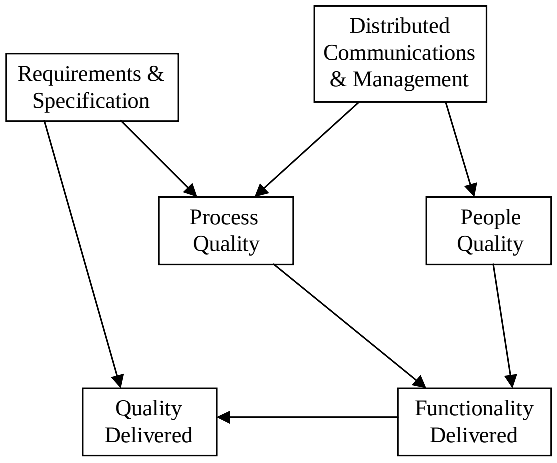


Figure 2. Schematic for the project BN

The subnets are:

- ***Distributed communications and management.*** Contains variables that capture the nature and scale of the distributed aspects of the project and the extent to which these are well managed.
- ***Requirements and specification.*** Contains variables relating to the extent to which the project is likely to produce accurate and clear requirements and specifications.
- ***Process quality.*** Contains variables relating to the quality of the development processes used in the project.
- ***People quality.*** Contains variables relating to the quality of people working on the project.
- ***Functionality delivered.*** Contains all relevant variables relating to the amount of new functionality delivered on the project, including the effort assigned to the project.
- ***Quality delivered.*** Contains all relevant variables relating to both the final quality of the system delivered and the extent to which it provides user satisfaction (note the clear distinction between the two).

The second reason for extending the simplified model of Figure 1 is to cope with variables that cannot be observed directly. Instead of making direct observations of the process and people quality, the functionality delivered and the quality delivered, the states of these variables are inferred from their causes and consequences. For example, the process quality is a synthesis of the quality of the different software development processes – requirements analysis, design and testing. The quality of these processes can be inferred from indicators, like the results of project audits and of process assessments, such as the CMM. Of course, only some organisations have been assessed to a CMM level, but this need not be a stumbling block since there are many alternative indicators. An important and novel aspect of our approach is to allow the model to be adapted to use whichever indicators are available. Some examples of this are given in the following sections, which describe two of the subnets in more detail.

4.1. People Quality

Figure 3 shows the variables and causal connections in this subnet, with connections to other subnets shown dashed.

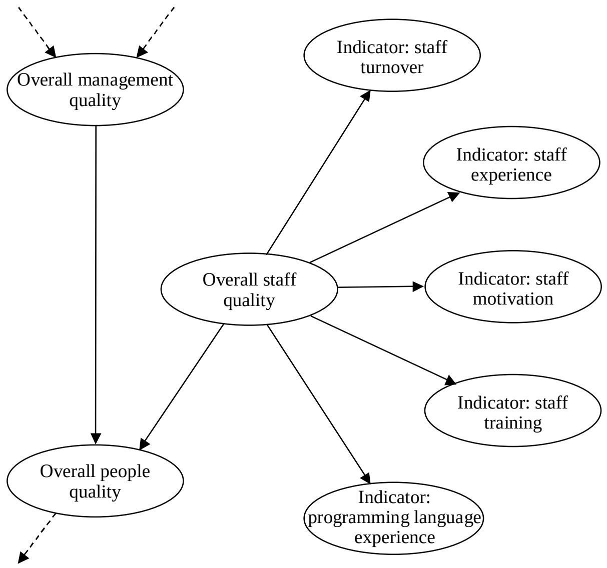

Overall management quality
Indicator: staff turnover
Indicator: staff experience
Overall staff quality
Indicator: staff motivation
Indicator: staff training
Overall people quality
Indicator: programming language experience

Figure 3. Subnet for people quality

A description of the main variables, including the model's rationale is given in Table 1. Variables, such as those described in Table 1, representing quality attributes have a 5-point measurement scale, ranging from very low to very high.

Observations are not normally entered directly on the variables described in Table 1. Instead we enter observation at primary causes (variables with no parents in the net) and 'indicators' (variables with a single parent and no children). Indicators can have either an ordinal or a numerical scale. Entering evidence at primary causes implies the use of deductive reasoning whereas observing an indicator leads to abductive reasoning, from effect to cause. Evidence propagation in a Bayesian net integrates deductive and abductive reasoning – the user does not have to distinguish between them.

Table 1. Details of subnet for people quality

Variable Name	Description
Overall management quality	This is a synthetic node that combines 'communications management adequacy', 'subcontract management adequacy' and 'interaction management adequacy'. If any of these three is poor then generally the value of 'overall management quality' will be poor.
Overall staff quality	This is the quality of non-management staff working on the project.
Overall people quality	This is a synthetic node that combines 'overall management quality' and 'overall staff quality'.

In the 'people quality' subnet (Figure 3), indicator nodes are used to infer the staff quality. The default variables in our model for this are: staff turnover, staff experience, staff motivation, staff training and programming language experience. In Section 6 we describe how a user can change these defaults to make use of the information available within a particular organisation.

4.2. Functionality Delivered

The 'functionality delivered' subnet is shown in Figure 4; each variable is described in Table 2. The values of variables such as 'project duration' are numbers – in this case any number ≥ 0. Since BNs require variables to have discrete states, this range is divided into intervals. To reduce discretisation errors it is desirable to have large numbers of intervals – up to 200 for some variables – and we are able to do this using the techniques described in [28].

This subnet incorporates the results of empirical research into software projects. For example, the so-called Brooks effect [6] suggests that effective effort is not a simple multiple of people and time because adding people to a project creates communication problems etc. This effect is factored into models like COCOMO and SLIM; it is represented in the net by a Rayleigh curve relationship whereby the overall effective effort increases at a lower rate as more people are added to a project.

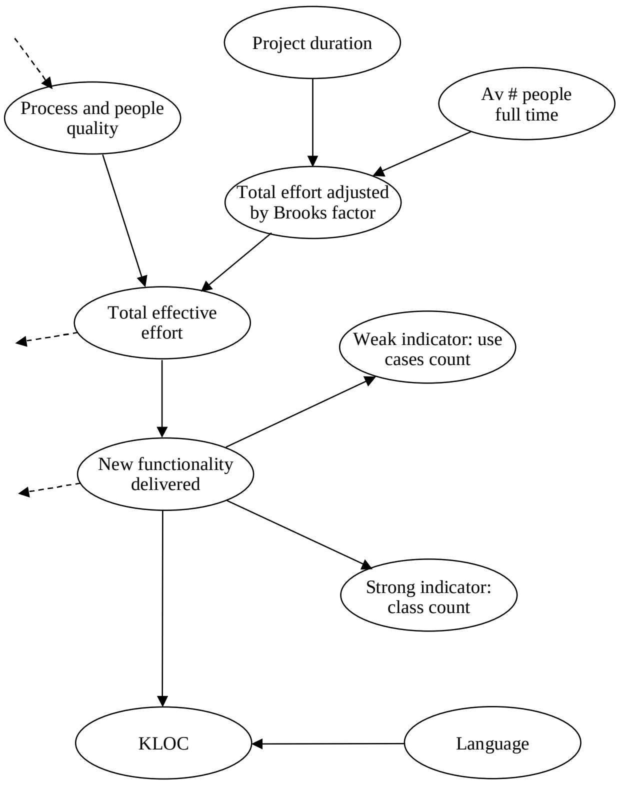

Project duration
Process and people quality
Av # people full time
Total effort adjusted by Brooks factor
Total effective effort
Weak indicator: use cases count
New functionality delivered
Strong indicator: class count
KLOC
Language

Figure 4. Subnet for functionality delivered

The amount of new functionality delivered is measured in function points [1, 31]. This is an attractive measure, since it is determined directly from the requirements and is independent of language and project phase. The relationship between the 'effective effort' and the functionality delivered (the number of function points implemented) is based on published data and models in [5, 17, 18, 20, 22, 24, 31, 32].

However, the data available does not cover the influence of people and processes factors although it is known that when these factors vary from best to worst, productivity (measured by functionality delivered for a given total actual effort) can vary by a factor of 20. This is represented in the net by an adjustment of effort to give 'total *effective* effort'. Experts were asked to judge the size of this adjustment influence, checking that the 'average' quality case is consistent with the empirical data.

In most cases users of the model would be expected to input an observation on the new functionality delivered variable and it might appear that this restricts our model to those organisations – regrettably few – using function points. But this is not the case, since the model allows you to enter observations via a number of indicators, the most obvious of which is Lines of Code (KLOC). The relationship between KLOC and function points is strongly dependent on the programming language used but we can use the extensive published empirical studies relating lines of code to function points in various languages (for example, [20]).

Table 2. Details of subnet for functionality delivered

Variable Name	Description
Process and people quality	Combination of 'overall process quality' and 'overall people quality' from the subnet concerned with people quality.
Project duration	Total elapsed time for software development work in the project in calendar months.
Av # people full time	Average number of people working full time on software development during the project.
Total effort adjusted by Brooks factor	The total effort in person-months, determined from the durations and the average number of people.
Total effective effort	An adjusted effort in person-months, taking account of the effect of process and people factors.
New functionality delivered	The amount of NEW functionality delivered, measured in function points
KLOC	The classic 'thousands of lines of delivered source code'.
Language	The programming language used in the system implementation.

However, there are apparent pitfalls of using KLOC as an indicator for the amount of delivered functionality in function points, since it is not a perfect indicator and this uncertainty is built into the model. If you believe you are delivering a 'big system' and enter a high KLOC value then the BN will, as you would expect, normally infer a high probability that FP is also high. However, this is not certain if you enter other conflicting evidence – a belief that the project will be completed to high quality, with a small team in a short time. In this case the BN may decide – quite rationally – that, irrespective of the high KLOC, the FP count is probably low.

The model also allows other indicators of the problem size to be used. There are many possible indicators, some specific to particular specification and development techniques (for example, the number of user scenarios specified using use cases or sequence diagrams) and other specific to particular project types (such as the number of database tables or the number of dynamically generated web pages).

5. Using the Bayesian net for Decision Support

What makes the Bayesian resource model so powerful, when compared with traditional software cost models, is that we can enter observations anywhere to perform not just predictions but also many types of trade-off analysis and risk assessment. So we can enter *requirements* for quality and functionality and let the model show us the distributions for effort and time. Alternatively we can specify the effort and time we have available and let the model predict the distributions for quality and functionality delivered (measured in function points). We show two examples of the model in use.

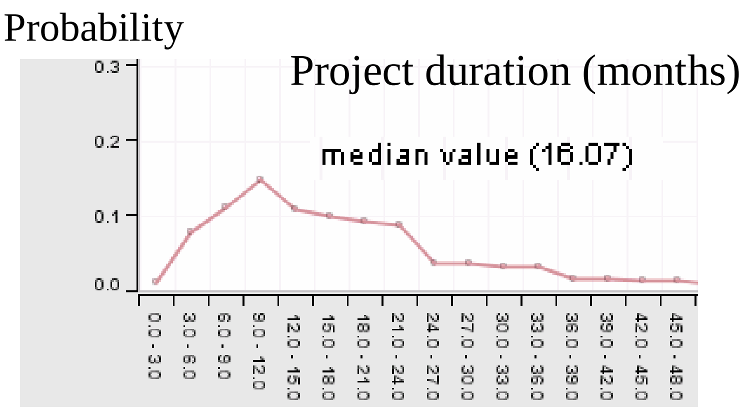
5.1. Resource Prediction

In this example we consider a 1000 FP (function point) project and compare the predictions made using our model with published data and other resource prediction models. We show that our model, in its 'dumbest sense' is consistent with published data and models, but also that it provides greater insight.

Entering a value of 1000 for the number of function points delivered results in the distributions for project duration and number of people shown in Figure 5 (this figure shows how probability distributions are displayed in the toolset we describe in Section 6).

Table 3. Predicted effort under various process and quality scenarios

Process & people quality	***Quality delivered***	***Predicted project duration months (median)***	***Predicted number of people (median)***	***Predicted total effort person months (median)***
high	poor	10	3	30
high	average	11	4	44
average	average	16	8	128
average	good	20	11	220
low	good	22	18	396
average	very good	25	24	600
average	perfect	29	29	841
low	perfect	40	42	1680



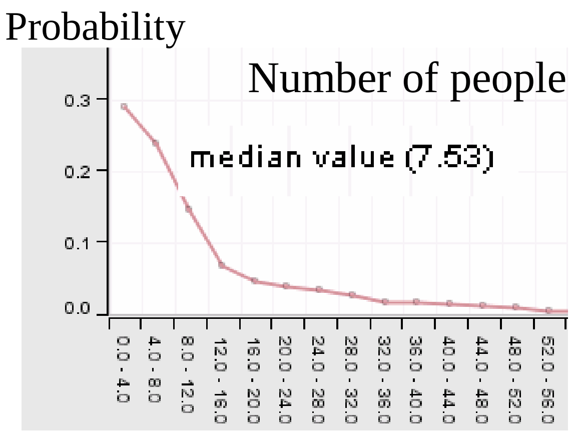


Figure 5. Probability distributions for project duration and number of people when FP=1000

Thus, the model predicts that the project should involve an average of about 8 people full time for 16 months. This is around 128 person months. However, because none of the other many model variables have values entered the distributions are, not very surprisingly wide. At this stage, for example, we know nothing about the quality of the product or of the people and processes. When we enter such observations, as in Table 3, we see a clearer picture of why the range is so large. For example, at one extreme if the process and people are consistently high quality and the delivered quality is poor then the model produces predictions whereby the median of the total effort is around 30 person months. At the other extreme, if the process and people quality is consistently low and the quality delivered must be perfect, then the total effort jumps to 1680 person months.

According to productivity rate ranges [21] based on data gathered from a number of completed projects, the total person months of effort for a 1000 FP project should satisfy the distribution shown in Table 4.

Table 4. Jones's productivity data for projects of 1000FP

% Projects	Effort
.01	< 10
.1	10-13
1	13-20
5	20-40
10	40-67
50	67-200
30	200-1000
4	>1000

So 50% of all such projects require between 67 and 200 person months of effort and 30% require between 200 and 1000 person months effort. It is difficult from this to interpolate the 'average' amount of effort but something like 150 person months seems likely. This is similar to our predictions in the 'average case'.

To compare the results in Table 3 with the most well known software cost model COCOMO we have to provide a KLOC estimate first. Let us assume the language is Ada. In this case other published data (for example Jones [21]) suggests a distribution of around 50KLOC for a 1000FP project (our own model produces a distribution with mean 51 KLOC). Using Boehm's basic model in organic mode the effort prediction is 146 person months – again this is comparable with Jones' data and our 'average case'.

It is also interesting to compare the people and schedule predictions using Boehm's schedule model. With this model a 138 person-month project should be completed in 17 months. Hence, in this case our model is an exact match with his. For a 396 person-month project (row 5 of Table 3), Boehm's model suggests a duration of 24 months, which is close to our prediction of 22.

5.2. Budget and Time Constraints

As a second example, we consider a fixed price contract in which we must deliver 2000 function points to perfect quality. Before entering resource constraints the median effort prediction in our model is 480 person-months (20 people for 24 months). However, we suppose now that the fixed price enables us to use just 6 people for 18 months. With these assumptions, Figure 6 shows the results for the distribution of *process and people quality* (the figure also shows the result when we relax the 'perfect quality' requirement). If we insist on perfect quality then the figure shows clearly that the quality of process and people almost certainly has to be 'very high'.

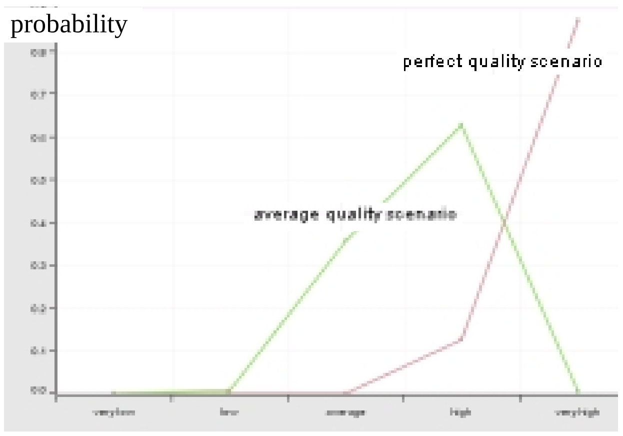


Figure 6. Process and people quality needed for an under-resourced project

Suppose, however, that we know that *process and people quality* is only 'average'. If we enter this as an observation then we can look at possible trade-offs we can make between functionality and quality. If we keep the 'perfect' quality requirement but remove the functionality delivered requirement, the model predicts a vastly reduced number of delivered function points (a median of 302 compared with the original requirement of 2000). If, on the other hand, we insist on the 2000 function points and remove the quality requirement the model predicts the quality as shown in Figure 7 (i.e. quality likely to be very poor to average, with only a very tiny probability that it will be above average).

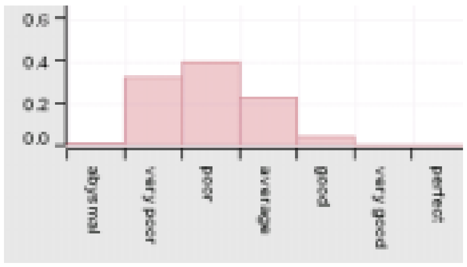


Figure 7. Quality delivered for fixed functionality in an under-resourced project

6. Validating the resource model in Agena's MODIST toolset

The Bayesian resource model described in Section 4 is used in the toolset being developed by Agena as part of the MODIST project [30] together with BN models for defect tracking and prediction. In this section we describe how a usable decision-support tool is built round the BN model and give initial results of trials of the tool and model.

6.1. Project Managers' Toolset

Our experience from earlier commercial projects such as [14, 25] is that project managers and other users who are not BN experts do not wish to use a BN directly via a general purpose BN editor. Instead, the BN needs to be hidden behind a more specialised user interface. The toolset provided by MODIST is actually an application generator that enables toolset users to tailor both the underlying BN models and the user interface that is provided to the end-users when the application is generated.

The main functions provided to the end-user are:

1. Observations can be entered using a questionnaire interface, where questions correspond to BN variables. Each question includes an explanation and the user can select a state (if the number of states is small) or enter a number (if the states of the variable are intervals in a numeric range). Answers given are collected into 'scenarios' that can be named and saved. At least one scenario is created for each software development project but it is possible to create and compare multiple scenarios for a project.
2. Predictions are displayed as probability distributions and as summary statistics (mean, median, variance). Distributions are displayed either as bar charts (see Figure 7) or as line graphs (see Figure 5) depending on the type of variable and the number of states. The predictions for several scenarios can be superimposed for ease of comparison (as shown in Figure 6). Summary statistics can be exported to a spreadsheet.

The application generator functions include the following:

1. The questionnaires shown to the end user can be configured widely. For example, questions can be grouped and ordered arbitrarily and the question text is fully editable. Not all variables need have a question, allowing any BN variable to be hidden from the end user.
2. Indicator variables used in the BN can be customised to match the indicators used within the organisation. As well as editing names given to an indicator in the questionnaire, its probability table can be adjusted. To make this feasible, the toolset provides a formula language with appropriate statistical distributions. We have found the truncated normal distribution useful for creating a probability expressing an expert's assessment of the 'strength' of an indicator. This approach, which is also used to adjust the probability tables of consequence variables synthesised from a number of causes, is described in detail in [25].

6.2. User Trial Results

The toolset is being trialled by various partners and user group members in the MODIST project. The main partner trials are taking place within Philips, Israel Aircraft Industries (Tel Aviv) and QinetiQ (Malvern). Philips is running trials at numerous locations including Bruges and Bangalore. Partners have validated earlier versions of the toolset and model, and provided feedback that has resulted in both model and toolset refinements. For example, at Philips projects are normally large and distributed but actually tend to be shorter compared to the 'heavyweight' projects that dominate much of the empirical software engineering data. Commercial pressures mean that projects that last longer than two years are not usually viable. Hence, there is a need for bigger teams working to shorter timescales. This kind of bias is now built in to the model and explains in part the slightly shorter and fatter predictions compared with the scheduling model in COCOMO. Perhaps more importantly than the numerical predictions themselves, the partners have found that the toolset enables them to reason in ways that was not possible before. This has helped them, for example, in rationalising 'go/no go' decisions about projects.

7. Conclusion

We have described a causal model for reasoning about the resources required for a software development project, using a Bayesian net. The results of empirical research into some key relationships present in our model, notably between the effort put into a project and the quantity and quality of software developed, has been used to build the net's probability tables. In this way, our approach builds on the results of existing research and could make use of future empirical research results in the same way. We have shown, however, that there are a number of advantages of incorporating the empirical relationships into a causal framework using a Bayesian network. Firstly, this allows the expert judgement to be used to supplement the data available. We believe that the lack of data is not just a temporary problem: software projects are too complex to allow relationships involving more than a few of the key variables to be investigated empirically. A complete statistical validation of the *joint* probability distribution of all the relevant variables can never be achieved.

Secondly, the causal framework provides flexibility to match the empirical data – using specific metrics such as function points – to the metrics used in each organisation. Thirdly, it provides much more flexibility in the way the data can be used to support decisions: predicting what can be achieved, what is needed to meet a target or how to trade-off the different outcomes in an under resourced project.

Although our approach is very flexible and does not insist on an organisation using specific metrics it does depend on some regularity in the way that development projects are estimated. This ensures that there are usable indicators for key attributes of quality and size, which our model can be adapted to use. It also makes it likely that the organisation's project managers will have accumulated the experience to be able to make stable judgments about the strength of these indicators. Some organisations may also accumulate data from past projects; there is no difficulty in principle in using such data in adapting the model and we hope to provide this capability in future versions of the toolset.

The trials underway in the MODIST project and with other partners provide the opportunity of further validation of the effectiveness of the decision-support available using the toolset we have described. We are also making available to the wider research community a version of the toolset [16] that incorporates the full underlying model described in this paper. This will enable researchers to investigate and validate the full model.

8. Acknowledgements

This report is based in part on work undertaken on the following funded research projects: MODIST (EC Framework 5 Project IST-2000-28749), .SCULLY (EPSRC Project GR/N00258), SIMP (EPSRC Systems Integration Initiative Programme Project GR/N39234), and SCORE (EPSRC Project Critical Systems Programme GR/R24197/01). We also acknowledge the contributions of individuals from Philips, Israel Aircraft Industries, QinetiQ and BAE Systems.

9. References

[1] Abdel-Hamid T and Madnick SE, Software Project Dynamics: An Integrated Approach, Prentice Hall, NJ, 1991.

[2] Abdel-Hamid TK, The slippery path to productivity improvement, IEEE Software, 13(4), 43-52, 1996.

[3] Albrecht AJ and Gaffney J, "Software function, source lines of code and development effort prediction", *IEEE Trans on Software Engineering*, SE-9 (6), 1983, pp. 639-648.

[4] Bangsø O and Wuillemin. PH, " Top-down construction and repetitive structures representation in Bayesian networks", In Proceedings of The Thirteenth International Florida Artificial Intelligence Research Symposium Conference, Florida, USA., 2000. AAAI Press.

[5] Boehm B, Clark B, Horowitz E, Westland C, Madachy R, Selby R, "Cost models for future software life cycle process: COCOMO 2.0", *Annals of Software Engineering*, 1995.

[6] Brooks FP, *The Mythical Man-Month: essays on software engineering, 2nd edition,* Addison Wesley, 1995.

[7] Fenton N, Krause P, Neil M, “Probabilistic Modelling for Software Quality Control”, *Journal of Applied Non-Classical Logics 12(2)*, 2002, pp. 173-188.

[8] Fenton NE and Neil M, “Software Metrics and Risk”, *Proc 2nd European Software Measurement Conference (FESMA'99)*. TI-KVIV, Amsterdam, ISBN 90-76019-07-X, 1999, pp. 39-55.

[9] Fenton NE and Neil M, “Software Metrics: Roadmap”, in *The Future of Software Engineering (Editor: Anthony Finkelstein) 22nd International Conference on Software Engineering*, ACM Press ISBN 1-58113-253-0, 2000, pp.357-370.

[10] Fenton NE and Neil M, “Making Decisions: Using Bayesian Nets and MCDA”, *Knowledge-Based Systems 14*, 2001, pp. 307-325.

[11] Fenton NE and Neil M, "Modelling risk in complex software projects using Bayesian Networks", RADAR Research Report TR104, 2003,

[12] Fenton NE and Ohlsson N, “Quantitative Analysis of Faults and Failures in a Complex Software System”, *IEEE Transactions on Software Engineering, 26(8)*, 2000, pp. 797-814.

[13] Fenton NE and Pfleeger SL, *Software Metrics: A Rigorous and Practical Approach (2nd Edition)*, PWS, ISBN: 0534-95429-1, 1998.

[14] Fenton NE, Krause P, Neil M, “Software Measurement: Uncertainty and Causal Modelling”, *IEEE Software 10(4)*, 2002, pp. 116-122.

[15] Fenton NE and Neil M, “SCULLY: Scaling up Bayesian Nets for Software Risk Assessment”, EPSRC Project GR/N00258 Final Report, Queen Mary University of London, www.dcs.qmul.ac.uk/research/radar/Projects, 2003.

[16] Forey S, Fenton NE, Neil M, Marsh W, “SCULLY toolset”, Queen Mary University of London, www.dcs.qmul.ac.uk/research/radar/Projects, 2003.

[17] Jeffery DR, The relationship between team size, experience, and attitude and software development productivity, Proc COMPSAC, IEEE Computer Society Press, 1987.

[18] Jeffery R and Stathis J, Function point sizing: structure, validity and applicability, Empirical Software Engineering, 1(1), 11-30, 1996.

[19] Jensen FV, *An Introduction to Bayesian Networks*, UCL Press, 1996.

[20] Jones C, *Programmer Productivity*, McGraw Hill, 1986.

[21] Jones C, *Applied Software Measurement*, McGraw-Hill, New York, 1991.

[22] Jones C, “Software sizing”, *IEE Review 45(4)*, 1999, pp. 165-167.

[23] Kitchenham B, Pickard LM, Linkman S, Jones PW, Modelling software bidding risks, IEEE Transactions on Software Engineering, 29(6), 542-554, 2003.

[24] Kitchenham BA, Empirical studies of assumptions that underlie software cost-estimation models, Information and Software Technology, 34 (4), 211-21*, 1992.

[25] Koller, D and Pfeffer, A. “Object-Oriented Bayesian Networks”, *Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI)*, Providence, Rhode Island, August 1997, pp.302-313.

[26] Neil M, Fenton N, Cates P, Forey S, Marsh W and Tailor M. “Modelling Subjective Causes and Objective Consequences in Bayesian Networks”, RADAR Technical Report 101, Queen Mary University of London, 2003.

[27] Neil M, Fenton N, Forey S and Harris R, “Using Bayesian Belief Networks to Predict the Reliability of Military Vehicles”, *IEE Computing and Control Engineering, 12(1)*, 2001, pp. 11-20.

[28] Neil M, Fenton NE, Nielsen L, “Building large-scale Bayesian Networks”, *The Knowledge Engineering Review, 15(3)*, 2000, pp. 257-284.

[29] Putnam LH, “A general empirical solution to the macrosoftware sizing and estimating problem”, *IEEE Trans Soft EngSE-4(4)*, 1978, pp. 345-361.

[30] QinetiQ, “MODIST Modelling Uncertainty in Distributed Software projects”, EC Framework 5 Project IST-2000-28749 www.modist.org

[31] Symons, CR, *Software Sizing & Estimating: Mark II Function point Analysis*, John Wiley, 1991.

[32] Stensrud E and Myrtveit I, Identifying high performance ERP projects, IEEE Transactions on Software Engineering, 2003.