Reliability Growth in Software Development Processes: A BBN Analysis of the Concept

Mario Brito[a*], John May[b]
[a]University of Southampton, National Oceanographic Centre, Southampton, England
[b]University of Bristol, Safety Systems Research Centre, Bristol, England

Abstract: In this paper we attempt to enhance process-based integrity argumentation, as used in software safety standards, to include a concept of reliability growth. Traditionally, the term reliability growth is used to mean the changes in reliability (or some other measure of quality) that occur during the debugging process. Here we use it to refer to changes in the quality of the wider software development process, as that process proceeds. The reliability growth phenomenon, in this new sense, is currently not addressed by software safety standards in any formal way, and its treatment in those standards is simplistic. This reliability growth is the result of complex dependencies in the quality achieved in different phases of the development lifecycle, and in particular by review processes, the effects of which reach across phases. In general, a review has the potential to influence integrity assessment for all processes prior to the review, including processes in earlier phases. We attempt to formalize this reasoning within a Bayesian Belief Network (BBN) model to encapsulate the deeper reasoning underpinning the notion of compliance to software safety standards such as IEC61508-3.

Keywords: Software reliability, Safety standards, Integrity claims, Bayesian belief networks.

1. INTRODUCTION

This paper brings together two strands of thinking in software reliability. The first strand is reliability growth, the observed improvement in dependability that typically occurs in test debug development lifecycle, [1,2,3,4]. The second strand is the process based prediction of dependability that is found in software safety standards. These standards use assessment of process quality to make software reliability claims, hence our use of the term reliability growth, or process reliability growth.

Software safety standards have industry acceptance. There are clear benefits to the process based approach they employ because it can be applied early within the development process cf. reliability growth testing. However, the reasoning in them is somewhat simplistic. When we attempted to formalize this reasoning we realised that insufficient account was being taken of the results of reviewing and testing. When we derived a model that could properly assimilate this evidence it became apparent that we were modelling a reliability growth phenomenon taking place throughout the whole development process [5].

In this paper reliability growth is added to the dependability argumentation used in software safety standards. Currently this is not addressed by software safety standards in any formal way. In order to illustrate how this could be achieved, the software development process was modelled using Bayesian Belief Networks (BBN). Use of Bayesian belief networks to predict software quality is now a well developed subject. The earliest work in the area appears to be in the FASGEP project, Hall et. al [6], which used BBNs to measure confidence in the software design process. This work included a simple model of reliability growth in processes, but although BBNs were used, a bespoke graphical probability model was needed to capture reliability growth. In this paper, we propose a more detailed model than that in [6], and one that is achieved entirely within the BBN formalism. Fenton and Neil in [7] present a critique of existing defect prediction models such as Multivariate models and use of size and complexity metrics, and conclude that BBNs offer some attractive benefits compared to the existing software metrics techniques.
This paper is organized as follows. Section 2 gives some background on software reliability growth models (RGMs). Section 3 explains our reasons for using BBNs for knowledge representation. Section

4 introduces the reliability growth phenomenon as an emergent property of the software development process. Our conclusions are presented in section 5.

2. RELIABILITY GROWTH MODELS

Software Reliability Growth Models such as the ones developed by Littlewood, Musa and Abdel-Ghaly in [1,2,3,4] are well known in the scientific community. Software reliability growth methods use time history of failures obtained during the software test debug phase to estimate important product metrics such as the number of initial faults, failure intensity, reliability, number of remaining faults, mean time between failures (MTBF) and mean time to failure (MTTF). A significant recent development in this field is reported by Bloomfield and Guerra in [8]. The authors applied a reliability growth theory named 'conservative theory' [9] to a software development process with the aim of supporting dependability arguments. They developed a model (the barrier model) in order to estimate the software reliability based on the number of errors found in the development lifecycle. The barrier model proposed by Bloomfield and Guerra was calibrated against data obtained from different software development projects. In their paper, the authors highlighted the lack of formalism in modelling the set of activities carried out during the software development process as one of the main reasons for discrepancies between number of failures predicted by the model and the number of failures detected in their case studies. Sections 3 and 4 of this paper highlight the benefit of using BBNs to model the software development process.

3. BAYESIAN BELIEF NETWORKS FOR ARGUMENTS REPRESENTATION

Bayesian belief networks provide a compelling paradigm to framework the set of activities carried out in the software development [10,11]. The appeal of BBNs lies in the strong foundations in probability theory, and the transparency provided by the graphical formalism. Bayesian Belief networks (BBNs) provide a powerful formulation to capture subjective arguments. As a Graphical Probabilistic model, BBN provides means for the experts debate the structure of their arguments visually [12]. Experts can assess visually which variables influence the domain and also the dependency conditions amongst variables. In the software assessment context, a variable may be described by a proposition such as 'confidence that the product meets the reliability target'. A BBN consists of a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ linked through causal connections. For such network the following probabilistic model can be derived.

$$P(V) = P(v_1,\ldots,v_n) = \prod_{i=1}^{n} P\left(v_i / \pi_{v_i}\right) \quad (1)$$

Where π_s are the parents of v_s. There is a one to one correspondence between the nodes in the network structure and the random variables that form the probabilistic model. In order to illustrate how BBNs can be used, consider the following typical example, where it is intended to estimate the ground slipperiness(G) given that we may or may not know the state of variables Cloudy(C), Rain (R), Sprinkler(S) and Ground Moisture (M). For simplicity we only consider these five variables and their dependency assumptions are captured in Figure 1. The model is not complete and it has been included here solely to show very briefly how BBNs capture undertain reasoning. Specifically: 1) What is required in order to build BBNs; 2) How BBNs provide belief update for any random variable presented in the model.

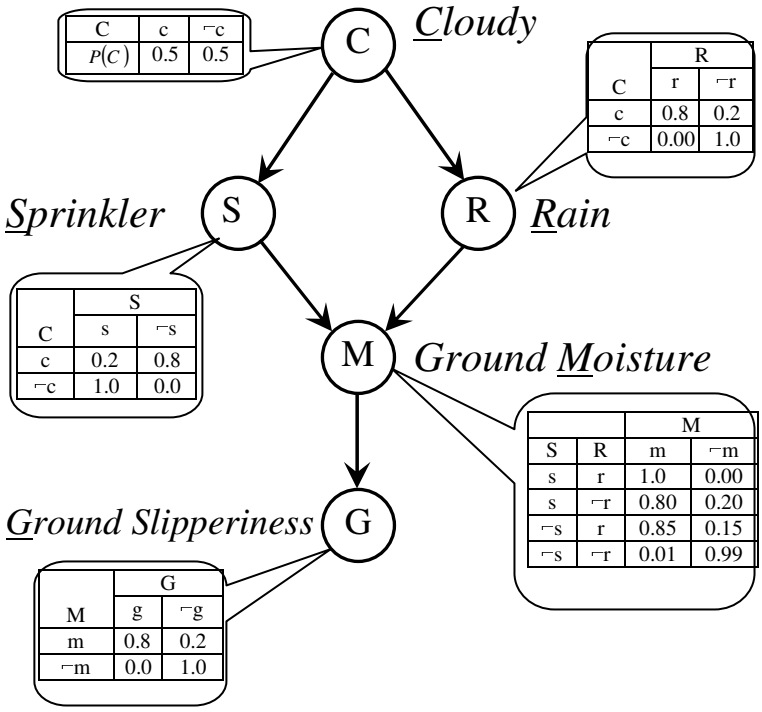

Fig1. Bayesian Belief network model for the Ground Slipperness model.

Where ¬c stands for the NOT c (not cloudy). The probabilistic model that corresponds to the BBN structure is as follows:

$$P(C,S,R,M,G) = P(C)P(S|C)P(R|C)P(M|S,R)P(G|M) \quad (2)$$

The belief that the ground is slipperiness can be computed by applying the conditional probability rule.

$$P(G = g) = P(G=g|\ M=m)P(M=m) + P(G=g|M=\neg m)P(M = \neg m) \quad (3)$$

Whilst the likelihoods correspond to the conditional probabilities that are specified in the respective network probability table (NPT) the marginal probabilities P(m) and P(~m) = (1 – P(m)) must be calculated according to the conditional independence conditions entailed by the structure. Given the NPTs, the probability distributions for node S and R will dictate the resulting probability distribution for the node M.

Much of the criticism about designing and using BBNs focuses on the definition of the conditional probability tables. Probability distributions present in a BBNs' conditional probability table often capture human beliefs or opinions about propositions. Assigning probabilities to events is based on expert knowledge; these are sometimes personal judgments that are otherwise unvalidated This knowledge may be elicited through either an informal or formal expert judgment elicitation exercise [13]. An example of a formal expert judgment elicitation process is presented by Otway and Winterfeldt in [16]. Otway's et al elicitation process consists of seven steps: 1) identification and selection of experts; and 2) training in probability judgments; and 3) presentation and discussion of uncertain events and quantities; and 4) analysis and data collection; and 5) Presentation and discussion of results of the previous step; and 6) elicitation; and 7) analysis, aggregation, and documentation. Keeney and Winterfeldt [14] followed a similar process to elicit expert knowledge with regard to the probability of catastrophic failure in two American Nuclear Power Stations namely Surry and Sequoyah following the individual or combined failure of two check valves that connect the

reactor cooling system and the residual heat removal system [14]. Winterfeldt elicited probabilistic values from approximately 40 experts concerning 50 possible events. A similar process was also adopted by Cockram in an influential PhD thesis [15]. He elicited probability judgments from 28 experts in order to populate his BBN NPTs. In addition to eliciting expert's judgment concerning the effectiveness of the software inspection process this work also conducted sensitivity studies on BBN models. One important issue that is often raised in context of expert opinion elicitation is how can one combine opinions from multiple experts? This is a field of research in its own right and publications on this topic can be found in the International Journal of Statistics or Knowledge Management, for instance. Methods that are most often applied are the 'linear opinion pool' and the 'logarithmic opinion pool' and significant developments have been made in developing techniques that are Bayesian based [16,17].

4. PROCESS BASED SOFTWARE RELIABILITY ARGUMENTS

Safety critical software reliability claims are guided by software safety standards, these standards relate a huge range of development choices to a reliability target [18,19]. These arguments are complex, opaque and cannot be deterministically modelled. Process models potentially inject transparency into this argument. Identifying a suitable process modelling approach is key. Given that the argument is by nature: 1) risk based, so probabilistic; 2) subjective; and 3) involving numerous inter-related factors, Graphical Probabilistic Models (GPMs) provide a potential solution, since they have the ability to capture all these facets. The following subsections are organized as follows; section 4.2 presents the network structure that captures the phenomenon of reliability growth in software development lifecycle. Section 4.3 addresses issues concerning its implementation in a BBN based tool.

4.2. Argument Structure in a Multi-phased Development lifecycle

The BBN network used to capture the software development process is presented in this section. The network structure was developed based on discussions held with the United Kingdom Health and Safety Executive. Interviews with two experts on standards based software assessment followed a semi-structured format where causal networks were used to manage discussions and organize the interview questions. In order to model the entire software safety development lifecycle a larger BBN is used to combine estimations from individual phases. This larger network feed forwards the quality of the development process of each single phase, since the subsequent development work will depend on that quality. The network also has feedback connections so that errors found in later phases have an impact on the contribution to the estimated SIL in a previous phase. This approach allows us to capture intricate influences between 'phases' in a way that goes well beyond the reasoning currently used in standards. The generic BBN structure shown in Fig. 2 contains a sub-net for each phase of the safety software life-cycle (not shown in detail) and a net for interaction among phases. The interaction aspect aggregates integrity estimates from multiple phases.

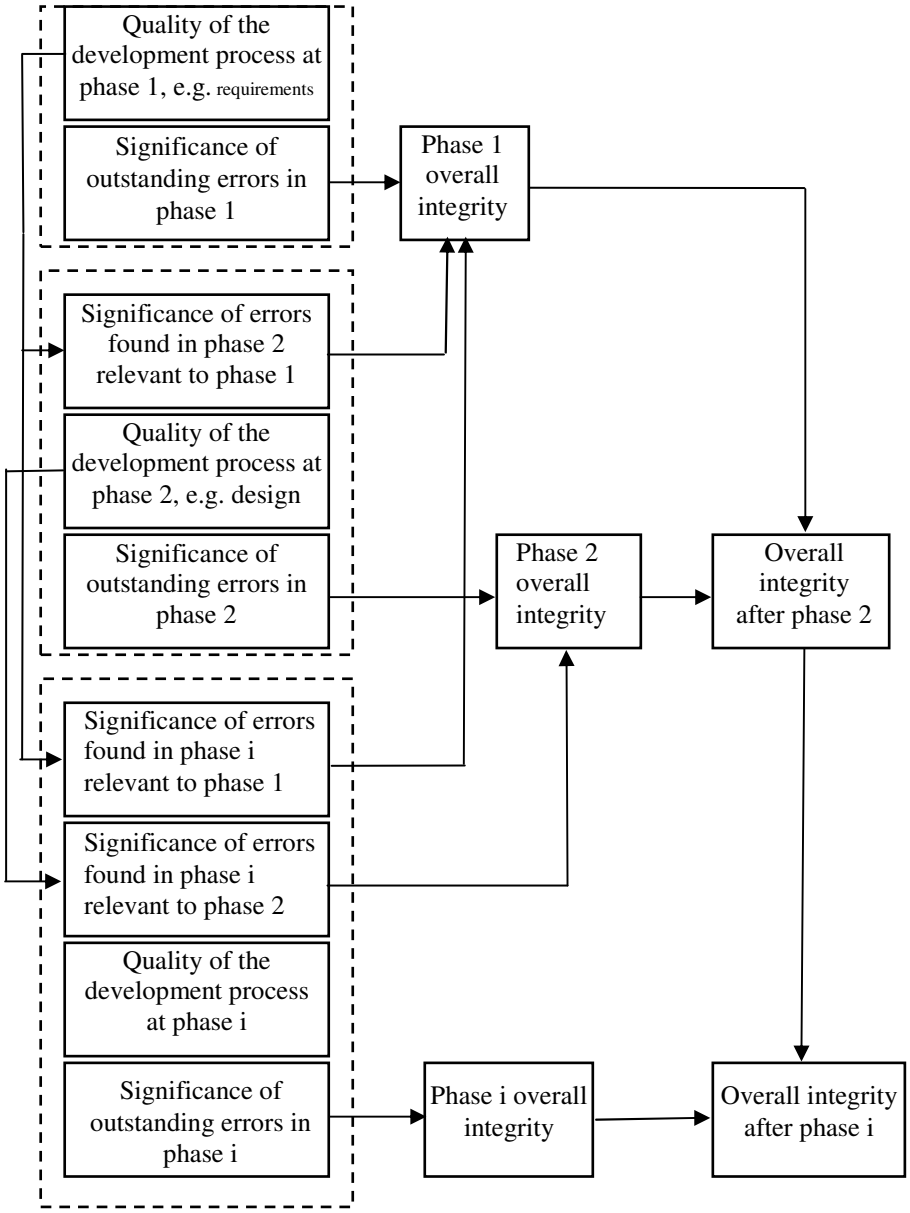

Fig. 2. Generic BBN Multi-Level structure for several phases of the safety software development lifecycle.

In each phase of the safety software development life cycle, there are verification exercises that aim to find errors introduced in the development process. This verification exercise, or process, certainly aims to find errors that are relevant to the particular phase at which it is being applied, but can clearly also find errors made in previous phases of the safety software lifecycle. Errors found in later phases are deemed to be corrected resulting in a gain in integrity level achieved at the end of the phase in which they were introduced (and hence subsequent phases too). An example is for instance, errors found whilst testing the software. Some of these errors will be relevant to the software implementation phase, some may have been introduced in the software functional specification phase.

We implemented a simple rule, that the overall integrity after any phase is the minimum level of integrity achieved for all previous phases including its own. For instance, if one claims SIL 1 for phase 1 and SIL 3 for phase 2, the overall integrity that one can claim after phase 2 is SIL 1. Clearly, this is a candidate for debate, and there is a strong case for additive models.

Bayesian belief network models of the software development lifecycle have also been independently proposed by Chin-Feng [10] and Fenton [11] and BBNs to model software development processes are gaining popularity in industry. The novelty of the BBN structure present in Figure 2 lies in its detailed modelling of reliability growth in software development processes.

4.3. Reliability Growth as an Emergent property in Software Development Processes

The reliability growth phenomenon is graphically illustrated in Figure 2. This phenomenon is made explicit in the causal feedback relation from errors found in later phases of the development lifecycle to the integrity claim that can be made for earlier phases.

The expert elicitation process of good NPTs is often a lengthy process of consensus building. We do not propose that the probabilities used in our model are correct. This notwithstanding it is still possible to experiment with the reliability growth phenomenon qualitatively using plausible and coherent assumptions are present in the NPTs. Table 1 depicts a fraction of the BBN conditional probability table for node 'Phase 1 overall integrity', using only three phases of the software development lifecycle are considered for this example. This fraction of the table captures the conditional probability distributions assigned for the case where the significance of outstanding errors in phase 1 is deemed tolerable. The first column of Table 1 is the significance of outstanding errors in phase 1 of the software development lifecycle – software requirements specification phase (A). In the second column of Table 1 is the significance of errors found in the design phase that are relevant to the requirement specification phase (B), in the third column in the significance of errors found in the system architecture design phase that are relevant to the requirements specification phase(C). All three nodes are given four states in this version of the model, {negligible, tolerable, undesirable, intolerable}. Row one in Table 1 captures the probability assignments given to the condition where A = tolerable, B = Negligible and C= Negligible. For this particular case the likelihood that SIL 2 can be claimed can also be written as P(SIL = SIL2 ¦ A= Tolerable, B= Negligible, C = Negligible) = 0.99. The values in Table 1 are used only to illustrate the practical aspect of implementing the model. The model as shown encodes the belief that more weight should be given to the errors found in the phase 2 of the development lifecycle. Errors found in phase 3 of the development lifecycle are not deemed as important as those found in phase 2 however they still have a positive effect on the belief that a target SIL can be claimed for phase 1 of the development lifecycle.

Table 1: Node Probability Tables

Significance of outstanding errors in the requirements specification phase	Significance of errors found in the system design phase	Significance of errors found in the architecture design phase	Phase 1 Overall integrity			
			SIL 1	SIL 2	SIL 3	SIL 4
Tolerable	Negligible	Negligible	0.00	0.99	0.01	0.00
		Tolerable	0.00	0.90	0.10	0.00
		Undesirable	0.00	0.80	0.20	0.00
		Intolerable	0.00	0.70	0.30	0.00
	Tolerable	Negligible	0.00	0.80	0.20	0.00
		Tolerable	0.00	0.70	0.30	0.00
		Undesirable	0.00	0.60	0.40	0.00
		Intolerable	0.00	0.50	0.50	0.00
	Undesirable	Negligible	0.00	0.70	0.30	0.00
		Tolerable	0.00	0.50	0.50	0.00
		Undesirable	0.00	0.40	0.60	0.00
		Intolerable	0.00	0.30	0.70	0.00
	Intolerable	Negligible	0.00	0.60	0.40	0.00
		Tolerable	0.00	0.40	0.60	0.00
		Undesirable	0.00	0.20	0.80	0.00
		Intolerable	0.00	0.10	0.90	0.00

5. CONCLUSION

The proposed Bayesian Belief Network model attempts to capture the phenomenon of reliability growth in software development processes. The need to model this phenomenon emerged from interviews with experts when we were attempting to formalize the reasoning underpinning compliance to software safety standards. There is a strong argument for formalizing the underlying reasoning in such safety standards using BBNs and indeed BBNs have previously been applied to software safety standards by Gran in [20]. However when Gran used BBN to model recommendations made DO-178 software safety standard Gran did not attempt to model the software development lifecycle in detail. Projects where BBNs were used to model the software lifecycle were also briefly discussed (see section 4.2) however as highlighted in section 4.2 none of previous attempts to model the software lifecycle capture the reliability growth phenomenon presented in this paper.

Our approach could be applied to any product development process. Whether one is designing a hardware or software based system the initial belief in the product quality for a particular phase will increase as we correct problems that were found in later phases.

We used a BBNs framework to model the emergent property of reliability growth however it would be interesting to see whether process models based on a systems dynamics approach would also enable us to quantify this property.

The BBN formalism provides a rich model for argumentation, that can also be imported directly into a software tool to support the application of software standards. Tools for implementing BBNs are commercially available. Tools such as HUGIN, NETICA, MSBNx, XBaies2, SEAMED, AGENA would allow the modeling of the proposed network.

As in any approach, BBNs also have their limitations. For many the probability assignment exercise seems to be their major weakness. However there have been developments in methods to elicit probability values from experts and also to aggregate opinions of several experts that aid this exercise. It is necessary that that both domain experts and knowledge engineers understand that this exercise requires considerable time and effort to prepare questionnaires and also to fill them in.
Another potential area of weakness for BBNs is that the conditional independence relations in the network structure are often unsubstantiated.

Acknowledgements

We would like to thank our sponsors the Health and Safety Executive (under contract number 6013/R38.039), the EPSRC, and Stirling Dynamics Ltd. We would also like to thank our two anonymous experts.

References

[1] I. Musa, Okumoto. Software reliability - Measurement prediction, application. ISBN0-07-044093-X. 1987.
[2] Musa, J.D.,1998. Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw Hill.
[3] Littlewood B. 1991. Software Reliability Modelling. Achievements and Limitations. Proceedings IEEE CompEuro91, Bologna, May 13-16 , IEEE Computer Society Press, 1991.
[4] Abdel-Ghaly A.A., Chan P.Y. and Littlewood B.. "Evaluation of competing Software reliability Predictions". IEEE transactions on Software Engineering, volume 12, issue 9, pages 950-967, September 1986. ISSN 0098-5589.
[5] Brito, Mario and May, John, H.R.: Gaining confidence in the Software Development Process using Expert Systems. Lecture Notes in Computer Science vol. 4166, pp 113-126.

[6] Hall, P., May, J., Nichol, D., Csachur, K., Kinch, B..: Integrity Prediction during Software Development. Safety of Computer Control Systems. (SAFECOMP'92), Computer Systems in Safety-Critical Applications, Proceedings of the IFAC Symposium,Zurich, Switzerland, 28-30 Oct 1992.
[7] Fenton NE and Neil M, 1999. A Critique of Software Defect Prediction Models. IEEE Transactions on Software Engineering 25 (5) 675-689.
[8] Bloomfield, R. E. and Guerra, S. 2002. Process Modelling to Support Dependability Arguments. In Proceedings of the 2002 international Conference on Dependable Systems and Networks (June 23 - 26, 2002). DSN. IEEE Computer Society, Washington, DC, 113-122.
[9] Bishop P.G. and Bloomfield R.E., 1996. A Conservative Theory for Long-Term Reliability Growth Prediction. IEEE Trans. Reliability, vol. 45, no. 4, December 1996.
[10] Chin-Feng Fan and Yuan-Chang Yu, 2004. Journal of Systems and Software archive, vol. 73, issue 2, pp. 193 - 203.
[11] Fenton, N. E., Neil M., Marsh W., Krause P., Mishra R.: Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets, submitted to ESEC 2005.
[12] Pearl J (1998). Probabilistic reasoning in intelligent systems. Morgan Kaufmann, San Mateo, ISBN: 0934613737, 1988 (revised 1997)
[13] H. Otway and D. Winterfeldt . Expert Judgment in Risk Analysis and Management: Process, Context, and Pitfalls. Risk Analysis, 12, 1, (1992).
[14] R. Keeney and D. Winterfeldt. Eliciting Probabilities from Experts in the Complex Technical Problems. IEEE Transactions on Engineering Management, volume 38, no. 3, 1991.
[15] T. Cockram. The use of Bayesian Networks to determine software inspection process efficiency. PhD Thesis. England, Open university (2002).
[16] Morgan, M. G., Henrion, M.:Uncertainty.: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis. Cambridge University Press. ISBN: 0521427444. (1990).
[17] Morris, A. P.: Combining Experts Judgments: A Bayesian Approach. Management ScienceJournal, vol. 23, no 7, March 1977.
[18] IEC61508 functional safety of electrical/ electronic/ programmable electronic safety-related systems parts 1-7. Published by the International Electrotechnical Commission (IEC), Geneva, Switzerland. 1998-2000.
[19] Smith, D., Simpson. K.: Functional Safety – A straightforward guide to applying IEC61508 and related standards. Elsevier (second edition), ISBN 0750662697.
[20] Gran, B. A.: Assessment of programmable systems using Bayesian Belief nets. Safety Science 40 pp 797-812. 2002.