May, J. H. R. (2008). On Evolutionary Algorithms for Multiple Criteria Decision Support in Bayesian Belief Networks Models of Dependable Software Development Processes.. 1. Paper presented at 35th ESREDA Seminar: Uncertainty in Industrial Practice - Generic best practices in uncertainty treatment, Marseille, France.

On Evolutionary Algorithms for Multiple Criteria Decision Support in Bayesian Belief Networks Models of Dependable Software Development Processes

Mario P. Brito
National Oceanography Centre
Southampton Waterfront Campus, European Way
SO14 3ZH, Southampton, England

John H.R. May
Safety Systems Research Centre, Bristol University
University walk
BS8 1TR, Bristol, England

Abstract

Recent research in software development process assessment and modelling has led to an increase demand for formalisms capable of providing reasoning under uncertainty. Such methods are used for providing decision support and build expert consensus when there is a huge degree of subjectivity. Researchers have argued that Bayesian belief networks (BBNs) is one of the most suitable formalism for this task. However, Bayesian belief networks have typically been used to allow the user to identify the most suitable software development process in light of one objective only; this is usually product quality or number of latent faults in the product. In fact, the current BBN formalism does not allow the user to identify the optimal process with respect to many objectives. In this paper we argue that multiple objective genetic algorithms (MOGAs) embedded with the BBN model of the software development process can tackle this limitation. The proposed Decision support system (DSS) searches for those solutions that maximize the confidence in the product integrity whilst minimizing the costs and the time taken to develop the product.

Keywords: Software Process assessment, management, Bayesian networks, evolutionary optimisation.

1. Introduction

Bayesian belief networks are maturing as a mathematical approach to support the development of tools for software process risk management [1],[2],[3],[4]. In the field of software dependability the use of BBNs has also been noted in major national and international research projects such as *FASGEP*, *Datum, SHIP, DeVa* and more recently, in the UK EPSRC INDEED project. The general approach for these projects uses the underlying assumption that errors are introduced during development and models of this process will allow the project manager to assess the level of the

problem and identify preventive measures needed in order to avoid the introduction of errors [5][6][7][8][9][10]. In the specific context of software safety standards Gran in used BBNs to model the requirements present in DO-178 software safety standard [11].
One type of decision support system based on BBNs is designed by adding utility and decision nodes to a Bayesian belief network model (pure BBNs contain chance nodes only) [12][13]. This type of network is said to form an Influence Diagram or Decision network [14]. This paper does not use these formalisms since they do not provide sufficient support for multi-criteria decision making. In light of this limitation we have proposed integrating the BBN formalism with multiple objective evolutionary algorithms to search for optimal decisions in multiple criteria decision problems [15].

This paper is organised as follows. Section 2 gives an introduction to BBN modelling and defines the model used in our problem. Section 3 presents the BBN decision support system proposed in this paper. Section 4 provides two examples illustrating the application of our DSS. Finally section 5 presents our conclusions.

2. The Bayesian Belief Network Formalism

A Bayesian Belief Network (BBN) is a graphical representation of a set of random variables (the *nodes*) together with directed interconnecting links (*arcs*). The arrow forming an arc indicates the direction of a causal relationship between *parent* and *child* [16][17][18]. Where their aim is to encode human knowledge, such BBN models can only be validated through evaluation, i.e. by testing whether predictions made by the BBN model match those of the human expert.

Methods borrowed from social sciences are usually applied throughout the elicitation process in order to reduce various forms of inaccuracy such as bias [16][17]. Verification of BBNs based expert systems can be performed by giving to the expert system unseen scenarios and seeing if its predictions match those of the human expert. In [18] Cockram illustrates how to validate a BBN based expert system though sensitivity analysis.

For the BBN discussed here, each node has a set of discrete states, either numeric or as ordered descriptions. At each node, a *conditional probability table* (CPT) captures the relationships between the states of the parent nodes and those of the child node. These conditional probabilities are assigned by experts – usually a domain expert helped by a knowledge engineer.

A BBN model is typically composed of three types of nodes; these are namely *target*, *intermediate* and *observable* nodes. Target nodes are nodes that represent the variables of interest, variables for which we want to compute a probability distribution. Observable nodes are used to represent variables that are measurable or directly observable. In our case, the intensity at which a technique was applied and experience of the personnel are examples of variables that could be captured with an observable node. Note that observable does not necessarily imply ‘easy to measure’. Intermediate nodes are often defined to help manage the size of the conditional probability tables. These nodes often add transparency to the problem by representing

hidden variables or highlighting hidden interactions between variables. Thus, Intermediate nodes are typically of a qualitative nature.

Figure 1 show how causal relations between observable, intermediate and target nodes representing variables of our problem domain were established in order to define a probabilistic model to predict the confidence at which an adopted software development process complies with IEC61508-3 software safety standard [19]. The model depicted in Figure 1 was implemented in Hugin [20][21]. There are two major tasks when building BBN models: 1) defining the network structure; and 2) defining the node probability tables. The latter are used to quantify the strength of the causal relations. Similar to any other statistical problem in order to quantify the relations between different nodes one must often make strong assumptions [22]. A development process model is often broken into phases.

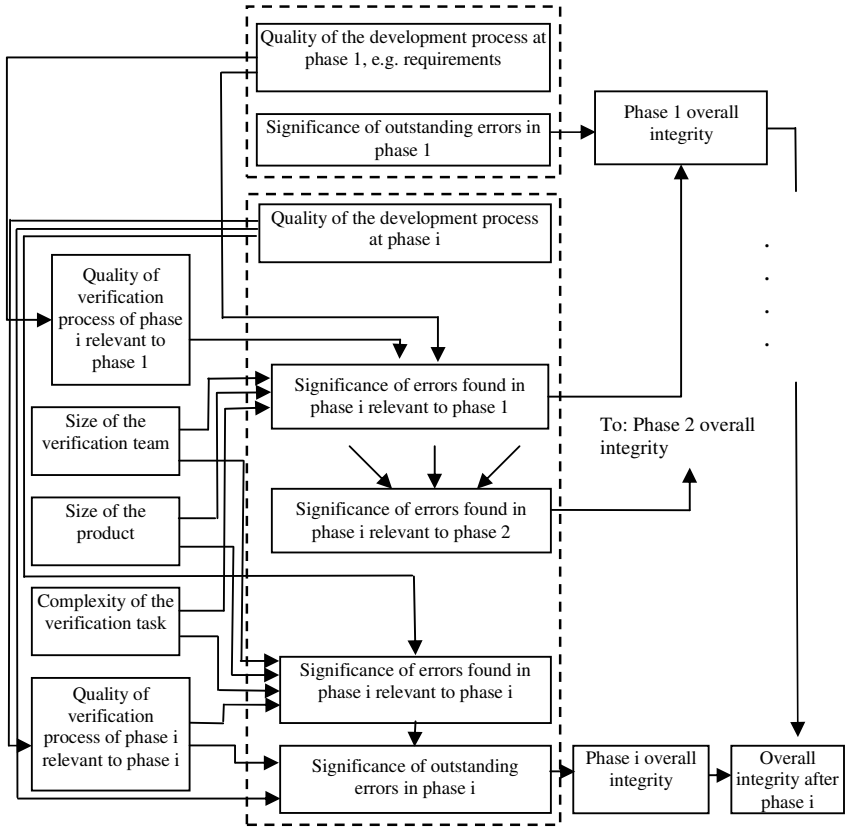

Figure 1. Generic BBN Multi-Level structure for several phases of the safety software development lifecycle.

The BBN in Figure 1 attempts to capture interactions between phases. Each dashed box on Figure 1 refers to a phase of the software development lifecycle. A detailed model for phase 1 (requirements capture) is presented in Figure 3. A process integrity level for each phase is estimated based on an estimated probability distribution of the significance of outstanding errors in that phase and also the distribution obtained for the criticality of errors found in later phases of the software development lifecycle relevant to the earlier phase.

3. The Decision Support System

The optimization algorithm was implemented in Visual C 6.0 and it communicates with the expert system through the Hugin Application Interface (API), [20]. The optimization algorithm collects 'rigid evidence' (this is evidence relating to facts that are fixed for a given project, captured in terms of specified values for BBN nodes) and runs "what-if" scenarios until it finds the most cost and effort efficient set. For instance, the algorithm can ask "what-if" I increase the intensity at which formal methods were applied, say from verifying a few key properties of the software requirements to verifying all required properties? Similarly, "what-if" we increase the number of the project review meetings? Given a set of user-specified fixed factors (constraints) the optimization algorithm will run all possible remaining scenarios in order to find the most cost and effort efficient solution or set of solutions. Figure 2 depicts the structure of the proposed system.

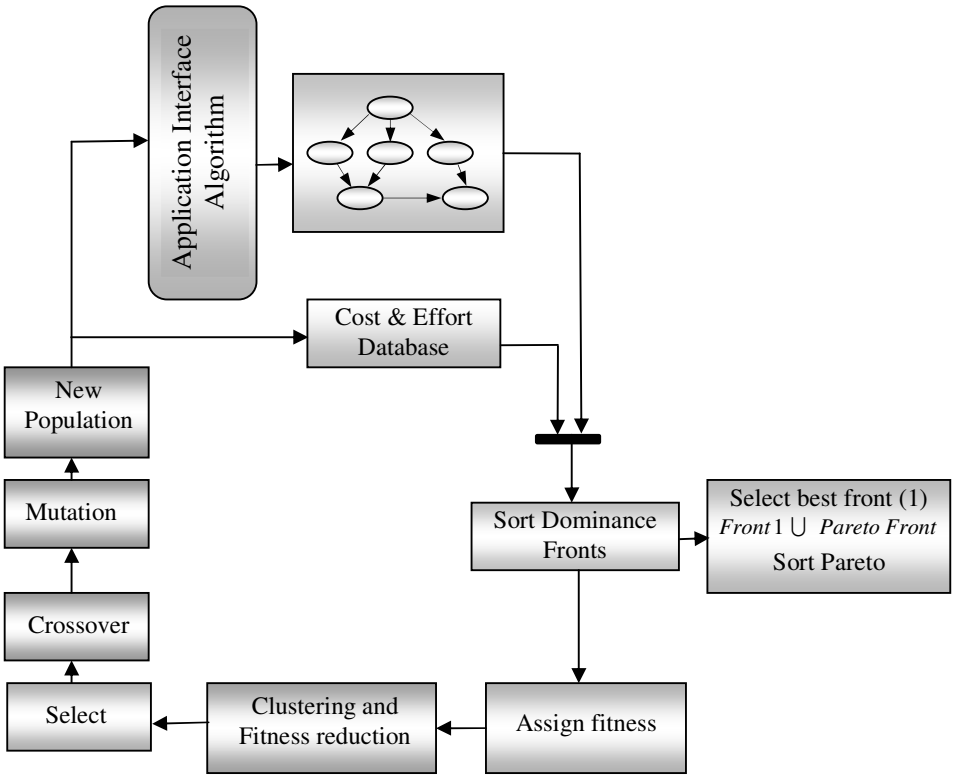

Figure 2. Framework of the general approach to risk management.

Given the constraints identified the optimization algorithm will run different scenarios. For each scenario the algorithm obtains the confidence in the SIL claim from the BBN model, and the cost and effort from a database. The optimisation algorithm operates binary strings that encode variable states. Each string aggregates the states for all BBN variables for which the state is known. New MOGAs are constantly being developed and their development usually involves large empirical studies. Convergence depends on the nature of the decision problem and also on the shape of the true Pareto front (whether the true pareto front is convex or concave). The NSGA algorithm is known for being able to outperform other non-elitist MOGAs

[23] [24]. However there are some challenges when it comes to integrate this algorithm with a BBN process model. For each generation the NSGA algorithm first sorts the GA population into different fronts; the first front contains all elements (or individuals or solutions) that are not dominated, the second front contains elements that are dominated by at least one element of the first front, and so on for the remaining fronts. The NSGA uses the Euclidean distance to measure the clustering amongst solutions of elements in the first front. This is part of a strategy that ensures diversity and spread of the optimal solutions. However the NSGA calculates the Euclidean distance based on the decision variables (e.g. input variables such as the 'intensity at which technique X is applied') rather than the values given by the objective functions (e.g. estimated process integrity level or costs or effort). BBNs usually have many decision variables. A plausible alternative would be to use the NSGA with a niching strategy that uses the Euclidean distance measured on the objective functions instead of the decision variables. The proposed algorithm stores all optimum solutions (non-dominated) on a separate set. So the algorithm contains two populations: one standard GA population where genetic operators are performed and another elite population containing all non dominated solutions found thus far. The standard GA population consists of 30 individuals. For the non-elite population, each individual is given a provisional fitness value according to its front, elements in the first front are multiplied by a fitness factor that reflects how close solutions are to each other. Thus solutions that are isolated are assigned a higher fitness level. Consequently these individuals will have a higher probability of being selected to create the next generation of individuals (solutions).

4. Using the Decision Support System

4.1 Two objectives: Optimisation of Integrity and Costs for the first phase of the Software Development Lifecycle

This case study analyses the software requirements specification phase of the software development lifecycle. It is assumed that the project manager has a clear idea of the size of the project and its complexity. The process constraints or, in other words, the variables that the DSS cannot vary to find the optimal processes are presented in Table 1.

Table 1. DSS input data case study 1

Attribute		State
Process constraints	Application factor	Moderate
	Complexity of the design	Fair
	Size of the verification team	Small
	Relevance of the verification method	High
	Complexity of the verification	Fair
Optimisation algorithm	Probability of mutation	0.08
	Probability of crossover	0.8
	Population size	30
	Elitism	No
	Chromosome size	30
	Number of Generations	300

Suppose the project manager now wants to know which development techniques to apply, the required competence of the staff and the type of verification technique that must be applied. The target node (the node whose probability distribution we are interested in) for this case study is the 'Phase 1 overall integrity'. We will consider two SIL targets, SIL 3 and SIL 4. SIL 4 is a highest level of integrity. It requires the application of better techniques.

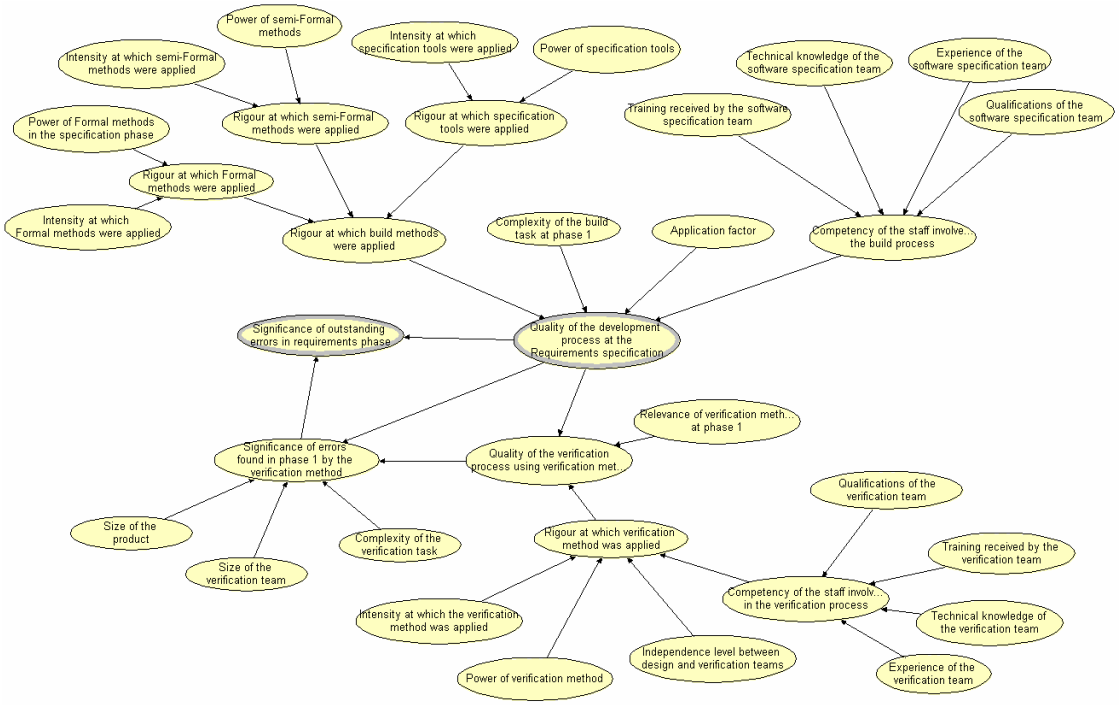

Figure 3 Hugin screenshot of phase 1 of the software development lifecycle.

Figure 4 presents the Pareto front obtained for this case study. Each data point represents a different process instantiation, i.e., a combination of techniques applied, the intensity at which they were applied and also the competence of the personnel involved in the development and in the review activities. Data point 1 in Figure 4 represents the cost-optimal process to follow in phase one in order to attain 83% confidence that SIL 3 can be claimed. The cost of the associated process is £2,300. In terms of techniques, this process (or scenario) involves the application of: a powerful formal method at a low intensity (in practice this might mean use of a formal method to provide a few key validated system properties, or maybe just the use of formal specification without any formal validation activities); a 'very good' semiformal method at a 'low' intensity; a 'moderate' verification method, such as a formal design review meeting at a 'low' intensity; development staff with satisfactory training and moderate qualifications, but lacking in experience (i.e. 'low' technical knowledge and 'low' experience); highly experienced verification staff (experience node was set to 'moderate' and technical knowledge node set to 'good'); and a high level of independence between the design team and the review team.

If the project manager wanted to have the same level of confidence (83%) that the software could claim SIL 4 instead of SIL 3 then he would have to follow the process corresponding to data point 2 in Figure 4. For this process, a 'very good' formal method was applied at a 'very high' intensity and a 'good' semi-formal method at a 'very high' intensity. The qualifications of the design staff are satisfactory and the experience and technical knowledge is 'high'. The verification activities followed in this process are identical to the verification activities followed in the process for data point 1. However the qualifications, training, experience and technical knowledge of the personnel involved in the verification process is 'high'. The independence level between the two teams is also 'high'. The process present in data point 2 is similar to the process present in data point 3. If one was to follow the process present in data point 3 then one would attain 96% confidence that SIL 3 could be claimed. On the whole the processes in data points 1 and 2 mirror the findings presented in industrial reports.

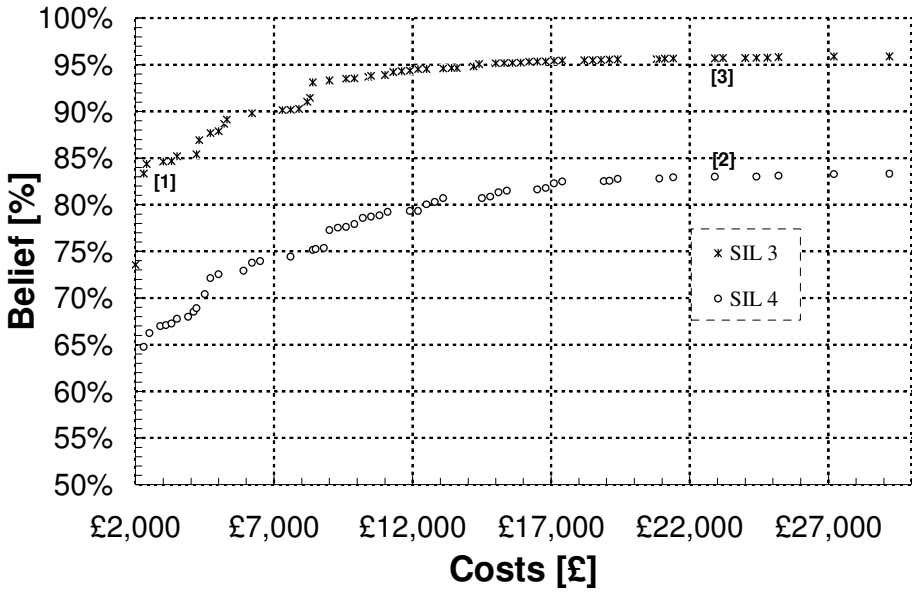

Figure 4 Non-dominated solutions after 2000 generations.

Concerning conformance to SIL 4, both Smith and Rivett in [25],[26] respectively argue that a formal specification should be carried out for the complete system, which in our example is addressed by the process represented by data point [2]. For SIL 3 however the two authors hold different views; whilst Smith argues that a semi-formal specification for the complete system, Rivett suggests that a formal specification should be presented for merely those functions that ought to meet SIL 3. In our example the optimal process (present in datapoint [1]) two techniques (formal and semi-formal specification methods) are applied at a low intensity.

4.2 Three objectives: Optimisation of Integrity, Costs and Effort for the first phase of the Software Development Lifecycle

In this example we consider the scenario where the user aims to find the optimal software development process with regard to three objectives: 1) belief that the target SIL (SIL 4) can be claimed; 2) costs of the software development process; and 3) the effort required by the process. The last two objectives are not conflicting by nature.

The initial assumptions concerning the software development process are identical to the assumptions presented in the previous example (Table 1). The Pareto front presented in Figure 5 contains the optimal processes for meeting SIL 4.

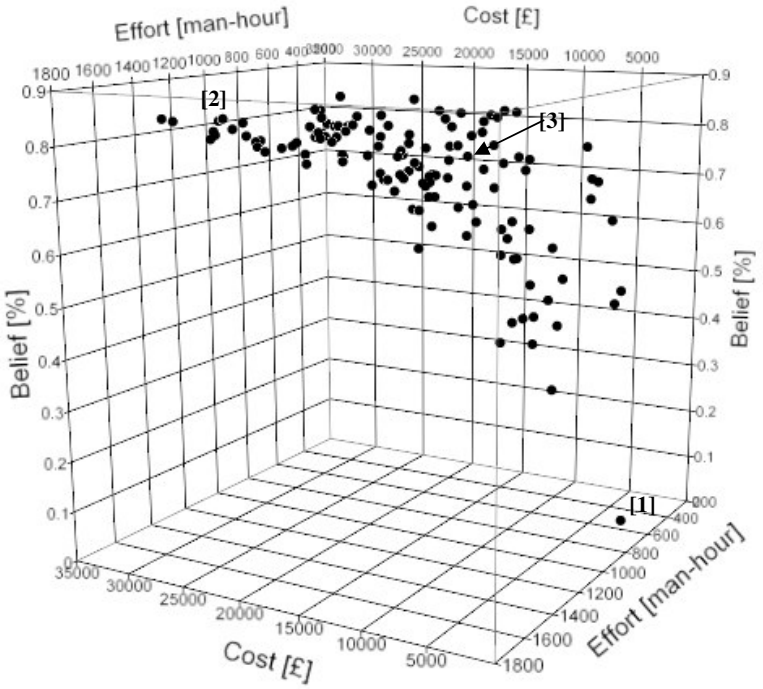

Figure 5 Non-dominated solutions after 2000 generations.

Data point 1 captures the process were a poor investment is made in terms of techniques used in the development. Both formal methods and semi formal methods were not used in this phase. Computer aided specification tools were applied at a high intensity. As result, the confidence that SIL 4 can be claimed is only 6%. This is the worst case scenario, it must be captured in the Pareto front because this captures the case where effort is minimised. On the other end of the spectrum the user may choose to adopt the process captured in data point 2. This process assumes that formal methods were applied at a very high intensity, semi-formal methods were applied at a high intensity and computer aided specification tools were applied at a very high intensity and no verification methods were applied. In addition the process uses highly qualified personnel with moderate experience. The process captured in data point 2 provides 83% confidence that SIL 4 can be claimed, it has a cost of £22,900 and effort of 1676 man-hour.

This method can be used by both the product developer and the person that is auditing the project. These two entities have different views on what is meant by an ideal development process. The auditor main concern is to maximise his belief that the target SIL is met while the product developer aims is to find the cost and effort efficient process that allows compliance with the target SIL. The process encapsulated in data point 2 is clearly not the ideal process from the developer perspective. A better development process for software product developer is the

process encapsulated by data point 3 gives 78% belief that SIL 4 can be claimed; it costs £14,600 and has an estimated effort of 1018 man-hours.

5. Conclusions

Bayesian belief networks have increasingly been used to framework expert knowledge in complex problems where there is huge subjectivity. These graphical probabilistic models have been used to support risk management and decision making in many industrial sectors, e.g. Nuclear, Military and Aerospace. An important aspect of any decision support system is that it should inform the user as to what is the optimal decision in light of a set of observations. Before our research this would only be supported using more or less classical approaches to utility theory. Bayesian network models that encapsulate utility theory (in the form of utility and decision nodes) are said to form an influence diagram. This method on its own is most suitable for problems where the user aims to optimise one objective only, say confidence in the product quality. Similarly to many BBN models, Influence diagrams can be large in size, with many utility nodes. In this case, finding the optimal solution for all objectives can be a tedious trial and error exercise, this effort increases exponentially with the number of nodes and the number of their states.

Developing safety critical software is often a costly and error prone process. The proposed DSS offers an interesting method to find a cost efficient set of techniques to follow in order to meet a target SIL. This is important information to support managerial decision-making regarding many key attributes, software product integrity and development costs, for instance, and their relationship. In one organization the project manager may be able to choose to increase the software safety integrity but will want to do so in as cost efficient manner as possible. In another organization, the project manager may choose to investigate whatever is possible in terms of integrity within a fixed budget, and use that to decide whether to go ahead with a project. The latter is a potential use of the tool in a contractual context, namely, to provide evidence to a purchaser that the required software integrity can be achieved at the quoted cost.

The examples used in the paper make use of notional figures. However, the method presented demonstrates that it is capable of capturing some of the rich relationships between quality and cost within the framework of development process modelling. The existence of this method can act as a means of establishing consensus amongst experts, both in terms of the structure of the model and in terms of the figures used.

With example one, we discussed processes present in the Pareto front obtained if one is targeting SIL 3 and SIL 4 for phase 1 of the development lifecycle. The results capture the simple idea that in order to have an effective and cost efficient process one ought to employ an experienced team to carry out review activities. Perhaps controversially, the BBN as built suggests that the experience of the personnel involved in the development process (for requirements capture) does not necessarily need to be high, provided that they have satisfactory training and good qualifications. This is clearly a point on which one might question the knowledge encoded in the BBN.

In the last example we looked at how the proposed DSS could be used to optimise three objectives. We considered the case where the project is at phase 1 of the software development lifecycle, and the target integrity level is SIL 4. The aim is to find those processes that will maximize our belief that a target SIL can be claimed whilst it minimizes the costs and effort. In many BBN applications the goal is to optimise a huge number of objectives. This is the case of the model proposed by Neil et. al [9]. Neil's model has nine utility nodes: maintenance costs, debugging costs, testing costs, assessment costs, design costs, assessment costs and benefits. The ideal process would optimise all these objectives. Similarly to our problem some of these objectives are conflicting, (e.g. for the same process an optimal design cost may lead to a poor assessment costs). However the model proposed by Neil et. al [9] does not address the problem of finding the best process with respect to all objectives. Finding the most suitable process with respect to all utilities would require a trial and error process that would take a long time, and in general, such an approach is not feasible in practice. We targeted this limitation in this paper.

Our algorithm converges quite quickly to the optimal solutions and this aspect has not been discussed in this paper. We may start to experience problems if we aim to optimise more than three objectives. The proposed DSS is based on genetic algorithms however it might be possible to improve the performance of the DSS using a different type of meta-heuristic optimization algorithm such as tabu search. Tabu search tackles an important issue in global optimization, namely, the multiple evaluation of a solution. Such algorithm might provide a faster trajectory to the Pareto front. There are many questions such as this remaining for further work. This paper provides a start in what appears to be a promising direction.

Acknowledgements

The authors would like to acknowledge the UK HSE for their insightful comments on the approach presented.

References

[1] Brito, M. and May J. (2006). Gaining Confidence in the Software Development Process Using Expert Systems. In Gorski J. (eds), Computer Safety, Reliability and Security; Proceedings of the 25th International conference on Computer Safety, Reliability and Security (SAFECOMP 2006), Gdansk, Poland, 26-29 September 2006. Lecture Notes in Computer Science 4166.

[2] Fenton N., Marsh W., Neil M., Cates P., Forey S. and Tailor M. (2004) Making Resource Decisions for Software Projects. Proceedings of the 26th International Conference on Software Engineering (ICSE'04), pp. 397-406.

[3] Chulani S., Boehm B. and Steece B. (1999) Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transactions on Software Engineering, vol. 25, no. 4, pp. 573-583.

[4] Madachy R., Subramanian, G. H. and Rodger, J.A. (2005) A Probabilistic Model for Predicting Software Development Effort. IEEE Transactions on Software Engineering, vol. 31, no. 7, 2005, pp. 615-624.

[5] Cottam M, May, J., et al (1994) 'Fault Analysis of the Software Generation Process - The FASGEP Project,' Proceedings of the Safety and Reliability Society Symposium: Risk Management and Critical Protective Systems, Altrincham, UK October 1994.

[6] Hall, P., May, J., Nichol D., Csachur, K. and Kinch, B. (1992) 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, October 28-30,1992, pp. 239-244.

[7] Fenton, N.E., Littlewood, B., Neil, M., Strigini, L., Sutcliffe, A. and Wright, D. (1998) Assessing dependability of safety critical systems using diverse evidence. *IEE Proceedings Software Engineering* 145 1, pp. 35–39.

[8] http://www.csr.city.ac.uk/projects/ship/ship.html.

[9] Neil, M. and Fenton, N. (1996) Predicting Software Quality using Bayesian Belief Networks. Proceedings of 21st Annual Software Engineering Workshop NASA/Goddard Space Flight Centre, December 4-5.

[10] Delic, K.A., Mazzanti, F. and Strigini L. (1997) Formalising a software safety case via belief networks, Proceedings DCCA-6, Sixth IFIP International Working Conference on Dependable Computing for critical Applications, Garmisch-Partenkirchen, Germany.

[11] Gran, B.A. (2002) Assessment of programmable systems using Bayesian Belief nets. Safety Science, vol. 40, pp. 797-812

[12] Fenton, N.E. and Neil, M. (2001) Making Decisions: Using Bayesian Nets and MCDA. Knowledge-Based Systems, vol 14, 2001, pp. 307-325.

[13] Fan, C. and Yu, Y. (2004) BBN-based software project risk management. Journal of Systems and Software, vol. 73, 2004, pp. 193-203.

[14] Burgess, C.J. Dattani, I., Hughes, G., May, J.H.R. and Rees, K. (1999) Using Influence Diagrams in Software Change Management. Proceedings of the 7th International Conference on Software Quality Management, March 1999, pp 177-187.

[15] Brito, M. and May, J.H.R. (2007) Optimization of Safety Critical Software Development Processes. Proc. European Safety and Reliability Conference (ESREL 2007), Stavanger, Norway, 25-27 June, 2007.

[16] Pearl, J. (1988) Probabilistic reasoning in intelligent systems, Morgan Kaufmann, San Mateo.

[17] Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L. and Cowell R.G. (1993) Bayesian Analysis in Expert Systems. Journal of Statistical Science, vol .8, no. 3, 1993, pp. 219-283.

[18] Lauritzen, S.L. and Spiegelhalter, D.J. (1988) Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. Royal Statistical Society Journal, vo.l 50, no. 2, 1988, pp. 157-224.

[19] IEC61508 (1998-2000) Functional safety of electrical/ electronic/ programmable electronic safety-related systems parts 1-7. Published by the International Electrotechnical Commission (IEC), Geneva Switzerland,.

[20] Hugin Expert A/S. (1990-2005) Hugin API Reference Manual version 6.4.

[21] Hugin A/S: http://www.hugin.com

[22] Littlewood, B. and Wright, D. (2007) The Use of Multilegged Arguments to Increase Confidence in Safety Claims for Software-Based Systems: A Study

Based on a BBN Analysis of an Idealized Example. IEEE Trans. Software Eng. 33(5): 347-365.

[23] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley.

[24] Deb, K. (2001) Multi-Objective Optimization using Evolutionary Algorithms, John Wiley and Sons Ltd.

[25] Smith, D., Simpson. K. (2004) Functional Safety – A straightforward guide to applying IEC61508 and related standards. Elsevier (second edition), ISBN: 0750662697.

[26] Rivett, R.S. (1997) Emerging Software Best Practice and how to be compliant. Proceedings of the Sixth International EAEC Congress.