Decision Support Software for Probabilistic Risk Assessment Using Bayesian Networks

Fenton, NE; Neil, M

Decision support software for probabilistic risk assessment using Bayesian networks

Norman Fenton and Martin Neil

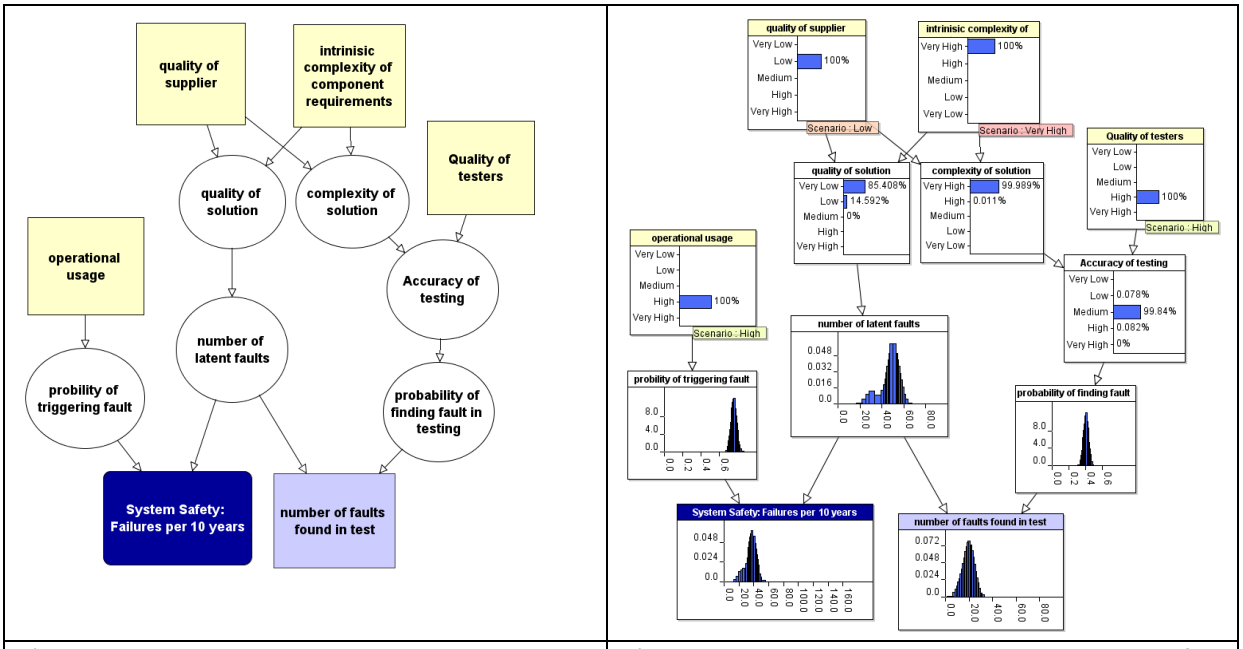
THIS IS A PRE-PUBLICATION VERSION OF THE FOLLOWING CITATION

Fenton, N. E., & Neil, M. (2014). "Decision Support Software for Probabilistic Risk Assessment Using Bayesian Networks". IEEE Software, 31(2), 21–26. http://dx.doi.org/10.1109/MS.2014.32

Why Bayesian networks?

Decision makers in all areas of life (including physicians, generals, scientists, bankers and politicians) must often assess and manage risk when there is little or no direct historical data to draw upon, or where relevant data is difficult to identify. The international credit crisis was not predicted by the world's leading financial analysts because they relied on models based on historical statistical data that could not adapt to new circumstances even when those circumstances (in this case the collapse of the mortgage sub-prime market) were foreseeable by experts with more intimate knowledge of the market place. The challenges are similarly acute when the source of the risk is novel: terrorist attacks, ecological disasters, major project failures, and more general failures of novel systems, market-places and business models.

Even though we may have little or no historical data, there is often an abundance of expert (but subjective) judgement, as well as diverse information and data on indirectly related risks. These are the types of situation that can be successfully addressed using Bayesian Networks (BNs) [2] , even when data-driven approaches to risk assessment are not possible. BNs (see Fig 1) describe "webs" of causes and effects, using a graphical framework that provides for the rigorous quantification of risks and the clear communication of results. The 'nodes' of a BN represent variables (that may or may not be observed and so are generally 'uncertain') while the 'links' represent causal or influential relationships.



a) The model structure

b) The model showing probability results after some observations are entered

Figure 1 Example of BN for assessing component safety

BNs can combine historical data with expert judgement, using calculations that are based on a simple theorem by the Reverend Thomas Bayes dating back to 1763. The theorem provides the only rational and consistent way to solve the problem of how to update a belief in some uncertain event when presented with new evidence.

Although Bayesian inference has been around for 250 years, it is only relatively recently that we have been able to actually use BNs for real-world problem solving. This is because the necessary Bayesian calculations are complex and quickly become infeasible. Indeed, it is know that exact probabilistic inference in BNs is “NP hard” [1] meaning that there is no efficient algorithm in the general case.

However, a dramatic breakthrough in the late 1980s changed things. Researchers published algorithms [7, 9] that provided efficient propagation for a large class of BN models. These developments were the catalyst for an explosion of interest in BNs. The first commercial tool to implement an efficient propagation algorithm was developed in 1992 by Hugin. Other BN tools quickly followed, such as Netica, Microsoft’s MSBNX, and BayesiaLab.

Improving the software support for BNs

During the 1990s we were working primarily on the problem of assessing the reliability of critical software systems [3]. BNs improved on our previous methods of assessment for this problem because it enabled us to incorporate expert judgement (for example, about the software development process) with limited data from individual component testing and even more limited data from system testing. We used the mainstream BN tools to build the necessary models to make predictions of fault and failure probabilities. Despite the difficulties of building such models their

enormous potential was obvious and Agena was set up in 1998 initially as a consulting company to exploit these opportunities.. Increasingly however, clients wanted to be able to use and adapt the BN models without permanent reliance on external consultants. Hence, more and more effort was spent on developing appropriate user interfaces to hide the underlying BN model complexity and access the underlying BN model(s) using the API of a standard BN toolset. Our experiences in building these types of system convinced us of the following:

- There were many fundamental usability problems with the mainstream BN toolsets that made it hard even for Bayesian experts to build serious models. None of the available toolsets handled continuous variables properly. This forced modellers to choose a static discretization of all such variables leading to unacceptable levels of inaccuracy.
- The commonality in the end-user GUI requirements of our clients was such that it ought to be possible to have a generic BN toolset. In other words, a different type of integrated BN toolset was required.

Hence, in 2004 Agena secured significant external investment, including bank finance, to develop its own BN platform and underlying inference algorithm (AgenaRisk), employing a team of full-time software developers for this purpose. Thus, Agena became a 'software plus consulting' company rather than a 'consulting plus software' company.

AgenaRisk development and deployment

Approaching this as software engineering professors, the development effort posed numerous challenges, both in the interfaces and the underling inference engine. It became obvious that an object oriented design was necessary to handle the complexity of modelling the generic BN structures. The additional need for a cross-platform solution suggested Java as a natural choice for the target language. However, we always knew that there was a critical trade-off involved with this choice. While it supported good OO design and portability, we had to work much harder to achieve acceptable levels of efficiency for the underlying highly computational intensive algorithms than would have been the case had we chosen C. Moreover, the choice of Java also conflicted with the extremely demanding and extensive GUI requirements.

We found that Java's built-in swing library, for example, was generally inadequate for most of our needs and we ended up having to develop many of our own GUI and graphing components from scratch to achieve the required look and feel (Fig 2). However, the extra effort was certainly worthwhile, since most users of the software compare it extremely favourably in this respect to other BN tools. Moreover, in 2011, a company paid Agena a significant sum to acquire the rights to use one of the GUI libraries we had developed.

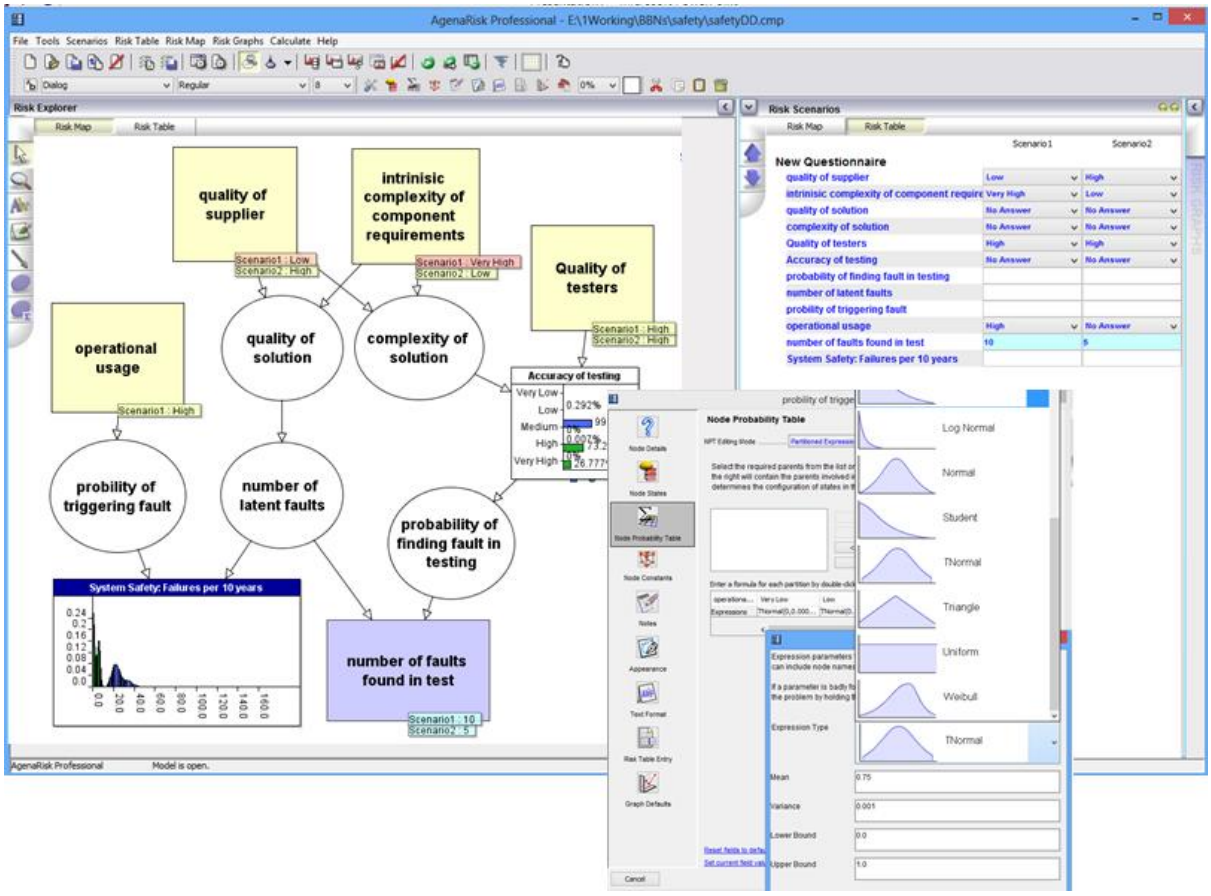


Figure 2 AgenaRisk GUI

We chose a lightweight version of UML and the Unified Process for design and development and this proved to be generally satisfactory. However, the pressure of commercial development was such that even a project partly managed by professors who taught software engineering was unable always to ensure adherence to the best of software engineering practices (see, e.g. [4]. For example, it proved extremely difficult to ensure a clean separation between the GUI code and the underlying BN modelling and statistical code, while there were never sufficient unit tests defined. However, the project has always maintained a rigorous online issues database (based on a tailored version of Bugzilla using a rigorous approach to fault and failure classification). The challenge for us, like any start-up company was that building the perfectly engineered product correctly would result in delivering it later than clients were willing to wait to pay for it. We have long term goals; they have short term needs. Software engineering – as taught in universities - ignores these commercial pressures and consequential trade-offs. The necessary difficult decisions then impact day-to-day choices: deliver a system with known but tolerable faults now, and get paid, or deliver highest quality code late, but go bankrupt.

The first commercial version of AgenaRisk was released in 2005. The marketing and pricing of this version was targeted at organisations with the same profile as those for whom we had previously developed bespoke solutions: defence, transport, banks, telecoms, and safety engineering companies; i.e. those who 'owned' systems that were critical in one way or another and for which quantitative risk assessment was necessary. Because of highly cited published research and models we had done in the area of embedded software defect prediction [Fenton et al 2007], this version of AgenaRisk sold most successfully to telecoms companies (most of the major players bought it) who

may have been especially attracted by the fact that the software came with an extensive set of fully documented reusable models. Most of the organisations buying AgenaRisk were doing so with the initial intention of developing a single model (sometimes with support from Agena consultants) to address a specific problem. In most cases the end users were just one or a small group of decision makers; however, sometimes the models were deployed more widely across an organisation (for example, one bank used their model for risk assessment across their full range of IT systems and this required multiple users across many sites to access the model).

Because of the sensitive nature of many of the applications for which AgenaRisk was being used it was extremely difficult to publicise the details of the applications, or to get public endorsements. However, in 2008, one of the major telecoms companies broadcast an internal company announcement that their use of a BN model in AgenaRisk – that enables them to manage the risks of replacing hardware components in the field, led to savings of $5 million per project. Descriptions of a large number of (mostly anonymised) case studies of the use of AgenaRisk can be found on agenarisk.com

Versions and licensing

There have been numerous subsequent releases of AgenaRisk (the latest being Version 6.0). The software growth in LOC is shown as Fig 3 and is comparable to (other refs in the series).

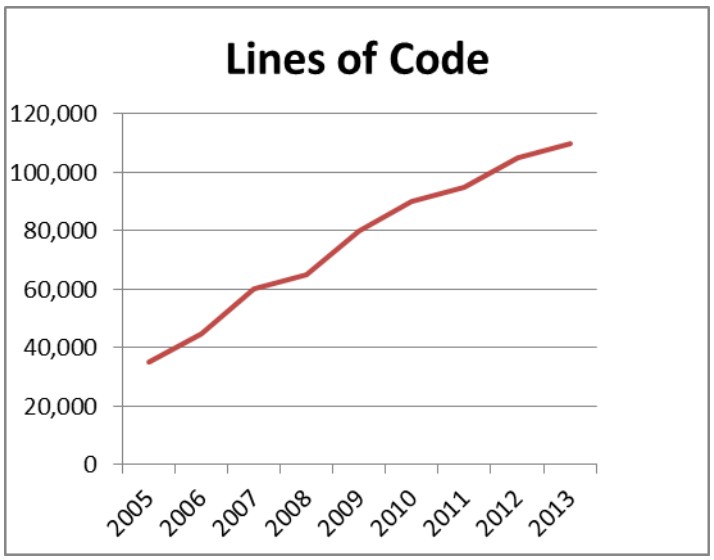


Figure 3 Increase in Lines if Code in AgenaRisk

In 2009, we released a version that implemented revolutionary algorithmic developments, such as the dynamic discretisation algorithm [8] and the ranked node method [6] that finally made it possible to easily build, respectively, accurate models containing continuous variables, and models involving ranked variables. By now we were targeting a wider audience and to support this strategy we were offering a free full evaluation version for one month that was downloadable automatically from agenarisk.com once a user completed a simple online form. This process was managed using the same licensing scheme as for the full commercial version (at that time the paid licenses were for lifetime use). The licensing scheme had been implemented by one of our own developers. While it was extremely effective in achieving its prime objective, for the end user it involved a non-trivial

process to install the licence key (the process varied, for example, with each different version of Windows because of different default admin settings used by Microsoft). This process worked fine for the commercial sales, but was ultimately unsuitable for the 'free trial' market, since we were inundated with requests for license key installation support despite the explicit instructions on the website.

With the release of Version 6.0 in 2012 we therefore changed our evaluation and licensing strategy as part of a switch in business model from perpetual licence to an annual subscription; while this impacted revenue in the short term our objective was to drive revenue growth in the medium term and support ongoing development of the software. Instead of an evaluation version we now provided a perpetual free version that required no license but had restrictions on saving models that used the most advanced features of the tool. This version could be downloaded from agenarisk.com without even the need for registering.

Fig 4 shows the cumulative number of registered downloads (i.e. it excludes downloads of the free version since 2012).

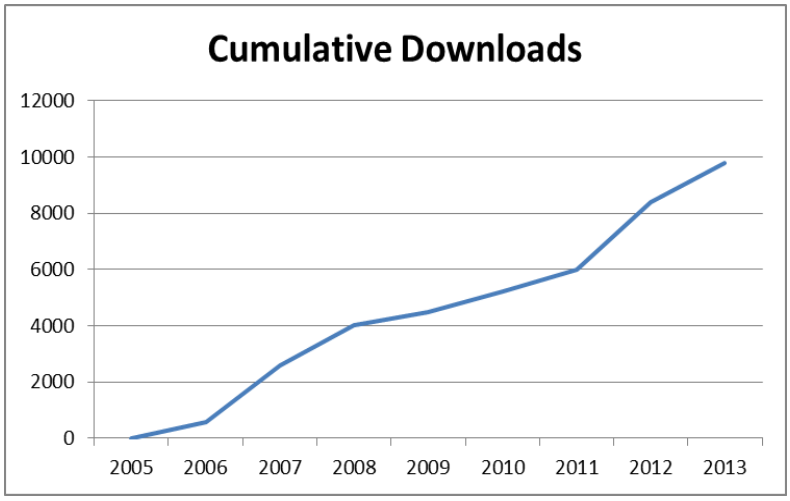


Figure 4 AgenaRisk Registered Downloads (free downloads since 2012 are not registered)

The future

AgenaRisk has made it much easier for non-statistical experts to build BN models to address serious risk assessment problems. However, there is still a long way to go before BNs truly 'cross the chasm' into mainstream business use. This means that the vast majority of business users continue to rely on decision support and risk assessment tools that do not provide the power, accuracy and insights of BN solutions. Agena continues to look for innovative methods of making both the development and use of BNs even more user-friendly. A web services deployment version is one obvious and attractive way forward; given the heavy processing that the algorithms demand of complex models, this involves addressing challenging problems of scale with multiple user access. However, it is clear that this is where the future lies.

References

1. Cooper, G. F. (1990). The computational complexity of probabilistic inference using bayesian belief networks . Artificial Intelligence, 42(2-3), 393–405.

2. Fenton, N.E. and M. Neil (2012), Risk Assessment and Decision Analysis with Bayesian Networks. 2012, CRC Press, ISBN: 9781439809105.

3. Fenton, N. E., & Neil, M. (1999). A critique of software defect prediction models. Software Engineering, IEEE Transactions on, 25(5), 675–689.

4. Fenton, N. E. (2010). Rational software developers as pathological code hackers. In J. Rost & R. L. Glass (Eds.), The Dark Side of Software Engineering: The Ethics and Realities of Subversion, Lying, Espionage, and Other Nefarious Activities (pp. 264–268). Los Alamitos: Wiley-IEEE Computer Society Press.

5. Fenton, N. E., Neil, M., Marsh, D. W. R., Hearty, P., Marquez, D., Krause, P., & Mishra, R. (2007). Predicting software defects in varying development lifecycles using Bayesian nets. Information & Software Technology, 49, 32–43

6. Fenton, N. E., Neil, M., & Gallan, J. (2007). Using Ranked nodes to model qualitative judgements in Bayesian Networks. IEEE Transactions on Knowledge and Data Engineering, 19(10), 1420–1432

7. Lauritzen SL and Spiegelhalter DJ, Local computations with probabilities on graphical structures and their application to expert systems (with discussion). Journal of the Royal Statistical Society Series 50(2), 157-224 (1988).
8. Neil, M., Tailor, M., & Marquez, D. (2007). Inference in hybrid Bayesian networks using dynamic discretization. Statistics and Computing, 17(3), 219–233.
9. Pearl J, Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241-288 (1986).