The KEBN Process: A new approach to Knowledge Engineering with Bayesian Nets

Andre Oboler
School of CSSE, Monash University
andre@csse.monash.edu.au

Abstract

Traditional and Object Oriented approaches to software development have models for the development life cycle of large software systems. Even when not followed in detail, life cycle models provide useful guidance to a complex task. The development of large scale Bayesian nets differs from other types of software development. These differences are a result of the Knowledge Engineering process needed for Bayesian Net development. Hence the lack of a software development approach that allows for Knowledge Engineering has had a significant impact on the uptake of Bayesian net technology.
This paper discusses some Knowledge Engineering approaches for Bayesian nets and presents the KEBN life cycle development model which accommodates the use of Knowledge Engineering and is specific to the domain of large scale Bayesian net development. The KEBN approach is compared to other traditional life cycle models and shown to be more compatible with large scale Bayesian net development.

1 Introduction

Bayesian nets have become one of the most popular and successful methods for artificial intelligence reasoning under uncertainty (Nikovski, 2000; Charniak, 1991). The networks are represented as directed acyclic graphs with conditional probability distributions in the nodes. They are also known as belief networks, causal networks, probabilistic networks and knowledge engineering maps (Charniak, 1991). Bayesian nets can be used for prediction, diagnosis, control, explanation, sensitivity analysis and calculating informational value. They are a clear and adjustable model of a system.

Knowledge Engineering is the acquisition, structuring and refinement of knowledge. The goal of knowledge engineering is to make information accessible to people or computer systems (Davidson, 1997). In the Bayesian net context, Knowledge Engineering is initially the acquisition of causal relationships and conditional probabilities from existing data and domain experts. Once established, Bayesian nets become the storage mechanism and structure for the knowledge. In all its uses, the Bayesian net either provides expert knowledge or uses that knowledge for expert quality system control.

Traditionally Bayesian networks have been limited in use due to their computational complexity. This has fueled research into more efficient algorithms. In recent times Bayesian net engines and computational power have both improved substantially. Bayesian nets are now in use in commercial and defense applications (Laskey and Mahoney, 2000; Haddawy, 1999; Charniak, 1991; Musman and Plehner, n.d.). The complexity of large Bayesian net systems

makes them difficulty to understand, build and maintain.

Users of Bayesian nets are struggling for a method to build and more accurately parameterise their software (Laskey and Mahoney, 2000; CoIL, 2000). Some leading researchers in the field have breifly refered to the area of knowledge engineering (Pearl, 1988; Heckerman, 1991; Jensen, 1996), but no methodology for the creation of large scale Bayesian nets has yet emerged. At present little research has been conducted in this area. Researchers still largely focused on algorithms (CoIL, 2000; Laskey and Mahoney, 2000). A software development approach that embraces Knowledge Engineering is greatly needed. One such approach is KEBN, Knowledge Engineering with Bayesian Nets.

We begin this paper with an introduction to Bayesian Nets in section 2. A brief overview of d-separation is given, this will be returned to later in the paper. Section 2 ends with possible uses for AI methods such as Bayesian nets. Knowledge Engineering is discussed in section 3. We consider issues related to expert solicitation and automated Bayesian learning with causal discovery algorithms. We consider factors in combining expert opinion and automated learning in a way that benefits both. The KEBN approach is introduced in section 4 and its various phases are explained. KEBN is then compared to existing lifecycle models, including the waterfall, spiral and prototyping approaches, which emphasises the suitability of KEBN and an approach centered on knowledge engineering.

2 Bayesian Nets

A Bayesian net is a DAG (directed acyclic graph) where meaning is given to both the nodes and the arcs between them. The nodes represent random variables. The arc typically represents a causal relationship in the direction of the arrow. The lack of an arc represents the absence of a direct dependency between variables. This can be seen in Figure 1.

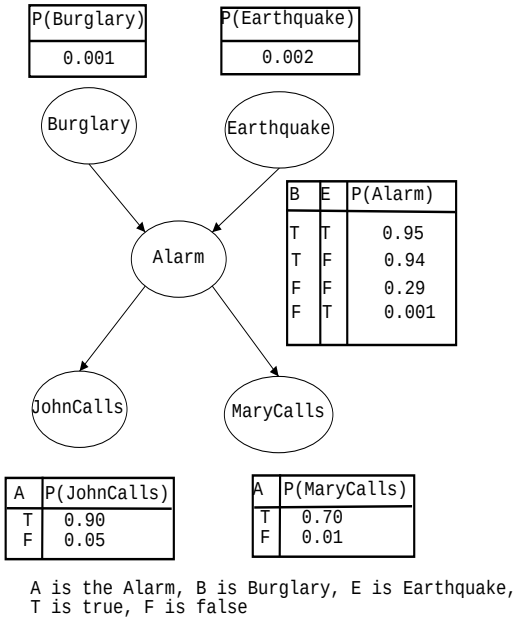

Figure 1: A simple alarm network. Adapted from Russell and Norvig, 1995

Figure 1 can be interpreted as: Burglary and Earthquake both have some probability, as given in the CPT (conditional probability table) for causing the alarm to sound. The alarm has some probability of causing either one or both of JohnCalls and MaryCalls. Note the meaning of the absence of an arc, the network models John and Mary calling based only on the alarm, they have no direct knowledge of Earthquake nor Burglary. Further knowledge from John and Mary such as noticing a tremor or seeing someone at the house, would require additional arcs to this network.

Independence and d-separation are key factors in fast Bayesian net update algorithms. D-separation may also be of useful in Knowledge Engineering, as discussed later in this paper. The three types of local structures found in a Bayesian net are shown in figure 2, the fourth

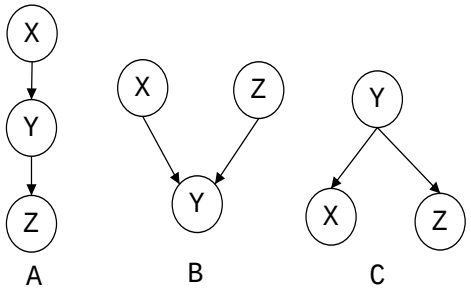

Figure 2: Types of directed connections for the undirected graph x-y-z

type is the absence of a arc. The structures in figure 2 are: a linear relationship (a), a converging one (b) and a diverging one (c).

Looking back at the alarm network we see both converging and diverging behavior. Consider the observation made above. Given the state of alarm, MaryCalls and JohnCalls provide no new information. This is a result of divergent behavior when the evidence (known or observed value) is the root. We can also see that convergent behavior tells us nothing if we know the value of the child node. i.e. given the alarm went off, learning there was an earth quake or burglary has no further effect on the probability that Mary or John will call. A linear relationship, like a, where the middle value is known, will behave in a similar fashion. These relationships segregate the network, given a known value in position Y. This property is called d-separation (Pearl, 1988).

2.1 Uses

While Bayesian Nets are only one of a number of approaches to data mining and decision support, they are one of the more versatile and weasily understood. Other approaches include: standard statistics, artificial neural nets, support vector machines, evolutionary and genetic algorithms, and decision trees (CoIL, 2000). The following tasks can be aided with one or more forms of decision support tools.A Bayesian Net can be used for any of them, and once constructed, can be reused for different tasks as needed.

2.1.1 Decision making

A Bayesian net can work out the optimal strategy for tackling some problem. A strategy is a list of decisions and which choices to take at each point. Optimal is the strategy that, on average, balances highest return (in money or utility value) and lowest risk. The optimal strategy may be tuned to return the maximum expected value (in the long run) or to be risk averse (favoring safer choices) like most people (Raiffa, 1968).

2.1.2 Forward Prediction

Given probability distributions of causes, a Bayesian net can calculate the probability of an effect occurring. For example the probability of a person developing asbestos poisoning given the parent probabilities such as general health, proximity to asbestos and duration of time in proximity to asbestos are given. Prediction gives the effect, given the cause (Nikovski, 2000).

2.1.3 Diagnosis

Given an effect, a Bayesian net can identify the most likely cause. For example, the likely cause of a plane crash may be calculated with a Bayesian net from the prior probability of failure of various components (including the pilot) and known facts about the plane crash. One of the most common uses of Bayesian nets is for medical diagnosis. The PATHFINDER project, later developed into a commercial product called INTELLIPATH is used at several

hundred medical sites (Heckerman, Horvitz, and Nathwani, 1992; Lam and Segre, 2002).

2.1.4 Control

Bayesian nets can be used for control using a combination of forward prediction (to work out the probability of an event occurring) and decision making (in order to work out the best course of action to achieve or avoid that result). In a control situation there is a feedback loop between the actions of the system and the resulting incoming data. The loop allows the net to adjust its beliefs and if needed adapt its strategy. Bayesian systems based on this principle have been developed for automation tasks as serious as the defense of American navy ships (Musman and Plehner, n.d.).

2.1.5 Explanation

Where a number of possible contributing causes exist, a Bayesian net can be used to calculate the most likely cause and the contribution of various causes. An adaptation of this is image classification as shown by a US Navy solution to the ship classification problem (Musman, Chang and Booker, 1993).

2.1.6 Sensitivity Analysis

A Bayesian net can be used to experimentally work out acceptable ranges for controllable variables. For example the effectiveness of hospital workers if their shifts are made slightly longer or shorter. The staff may be unaffected by changes under a certain size, or may drastically improve their effectiveness with a small reduction, while further reduction beyond that may have no effect. Sensitivity analysis allows the parameters to be unaltered until a change is observed.

2.1.7 Informational value

Like sensitivity analysis, information value may be worked out by trial and error adoption of the Bayesian net. Precision is a change in the degree of measurement. A lower degree of precision is easier and often cheaper to measure and maintain. In manufacturing and medical work it can lead to significant savings.

3 Knowledge Engineering

Knowledge Engineering is used both in the development and daily use of large scale Bayesian networks. Initially it is used to collect data and build the Bayesian network. Later the Bayesian network is used as a tool to store and obtain knowledge. The initial problem of building the Bayesian network is addressed first. Both expert knowledge solicitation and Bayesian learning by way of causal discovery algorithms will be discussed. We will then discuss combining the approaches. In the context of the combined approaches we consider methods to aid in obtaining knowledge from a Bayesian network. Though not specifically addressed the daily use of the Bayesian net may also benefit from the approach to combined discovery.

3.1 Knowledge Elicitation

Knowledge elicitation is the process of codifying the knowledge and decision making "rules" of human experts. Unlike rule based expert systems, real experts seldom make hard and fast decisions. The use of probability in a Bayesian net takes account of this uncertainty while causal relationships model the experts logic. The difficulty is firstly in obtaining the structure of the network and secondly in parameterising it.

The initial step in creating the structure of the Bayesian net is identifying the key domain variables (Druzdzel and Van der Gaag, 2000; Russell and Norvig, 1995). The next step is soliciting the causal relationship between variables. It may be done in one step with the expert providing a causal map, or multiple steps involving obtaining and ordering initially undirected arcs. Either way, it may require "significant effort" (Druzdzel and Van der Gaag, 2000). One approach to this problem is to pick only the those variables effecting (in the expert's opinion) the core of the problem. A Bayesian net development from a limited set may perform significantly worse than the expert when complications arise, but it is a good start for further development (Laskey and Mahoney, 2000). Once the basic system has been built, additional nodes and arc can be added as they are recognised or refinement is required.

Parameterising the network involves probability elicitation. Literature in this area dates back to the 1970s, and much of it is now dated (Laskey and Mahoney, 2000). An expert's knowledge is hard to access. Sometimes it contains significant inconsistencies that they may not be aware of, or not willing to acknowledge (Monti and Carenini, 2000). Some approaches have been developed to more accurately solicit probability. An early approach involves the use of "bets". In the betting or lottery method, the expert is asked to wager for or against a proposition. The cost of winning and losing are adjusted until the expert is indifferent to the choices offered. A version of the lottery method invovles two lotteries each having a major and consolation prize. The first lottery is based on an event X actually occurring. The second is based on X occurring with probability p. The expert is asked to choose which lottery they prefer. The value of p is adjusted until they are indifferent to which lottery they participate in (Keeney and Raiffa, 1976). Another method is an analytical hierarchy process, a ranking solicitation process adapted to probability solicitation (Monti and Carenini, 2000). Methods for more accurate probability solicitation are still being investigated.

3.2 Bayesian net learning algorithms

Causal discovery algorithms such as CaMML (Causal discovery using MML) (Wallace and Korb, 1999) exist and can create likely models from samples of collected data. CaMML works by sampling possible models. Gibbs sampling (Geman and Geman, 1984) with an MML estimators (Wallace and Korb, 1999) is used.

One problem with learning algorithms is the incorporation of new with old data. This can be dealt with using a time decay function for the influence of data (Kennett, Korb and Nicholson, 2001). Another problem is that data may hold bias that adversely effects the network constructions (Druzdzel and Van der Gaag, 2000). It may be hard to spot bias in the data. The CaMML system has been tested against a rule based systems created from expert solicitation. Without human intervention the models it produced consistently gave better predictions than the expert system (Kennett et al., 2001). This gives some hope that bias in the data may be less than the experts own bias.

3.3 Combined Approaches

We begin this section with a word of caution. Combining information from different sources can be risky. In the worst case it can lead to significantly worse results than either method used on its own (Druzdzel and Van der Gaag, 2000). The worst case occurs when the sources used have differing assumptions about the causal structure (Druzdel and Diez, 2000). If different

sources do not consider and adapt to each others views they may present incompatible results. While learning algorithms can be forced to take account of an expert's view, for best results the expert must also be willing to reconsider their ideas in light of the models "discovered" structure. This requires a clear understanding of the systems model by the domain expert.

One method of integrating expert knowledge with CaMML is to allow the expert to set prior probabilities on as many possible arcs as they wish. This has been implemented in a version of CaMML (O'Donnell, 2001). The MML scoring metric used in CaMML will reward models that match the experts opinion and punish those that disagree.

Ideally though, the expert should be able read and interpret the causal model. The experts knowledge should change and grow with the systems. The CoIL (Computational Intelligence and Learning Cluster) Network, an EU sponsored network of excellence, ran a data mining competition in 2000. Forty three groups from around the world, from both academic and industry, submitted a solution. CoIL organisers found that, "typical problems [with the solution approaches] are the focus on algorithms instead of methodologies and the lack of tools and efforts to explain the discovered patterns" (CoIL, 2000). As Peter van der Putten, one of the CoIL competition organisers observed, in real world situations prediction only models are not accepted by users. They do not contribute to the growth of knowledge about the problem (CoIL, 2000).

One useful abstraction in a Bayesian net is the grouping of d-connected parts of the network. This allows changes to the evidence to show up clearly as different parts of the net become d-separated and split themselves off from other groups. The property of d-separation is already used to optimise Bayesian net engine performance, as discussed earlier int his paper. It improved update times by limiting the amount of information the engine needs to process. In the same way is could reduce the complexity that domain expert needs to handle at any one time.

At the more local level, an equivalent of Boolean logic gates has been shown to prevent over fitting. For example, in a medical diagnosis, symptom 'A' (of which there are a number of types) can provide supporting evidence of disease 'MI' but on their own a number of 'A"s could lead to over confidence in a diagnosis of 'MI'. If the relationship is not just causal but mathematically logical, an "or" condition may be used to ignore repeated 'A' events. Nikovski, who suggests this, introduces an extra node to act as the gate (Nikovski, 2000). Rather than an extra node, adding to the complexity the human user must deal with, we suggest the entire gate structure could be modelled as a single (special) "node" on the network. This logic node could take many inputs and give 1 or more outputs, but instead of containing a conditional probability table, it would contain an appropriately structured sub-net. The sub-net could be hidden or expanded out at any time with no loss of information. This is shown in Figure 3). A similar construct (with the name ALL) could be used for AND. The names ALL and ANY have been chosen to convey meaning to the domain expert.

The idea of collapsible components relates back to structured analysis and would be familiar to many from techniques such as data flow diagrams (DeMarco, 1979).

Knowledge Engineering can be extended to a mutual learning process for both the network and the expert. Some considerations when taking this approach include:

- Ability to collect information about the world from various sources

- Ability to incorporate this information into the network's model of the world
- Visibility of the impact new information has on the model's structure, paramaterisation and performance
- The ability for the model to be understood by domain experts in the field being modeled.

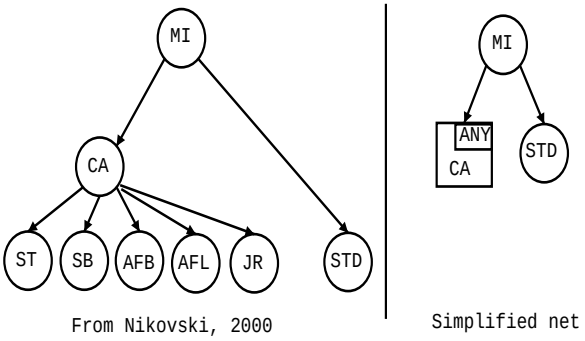

Figure 3: A full and simplified OR (ANY) component

Knowledge Engineering is a key part of large scale Bayesian nets development. It must be incorporated into any development methodology applied to Bayesian nets.

4 The KEBN Life Cycle

The KEBN life cycle (Korb, 2002), allows for the initial development of a network, followed by a number of phases (possibly repeated) leading to a Bayesian network in industrial use, and undergoing regular refinements. The KEBN lifecycle model is displayed in Figure 4.

In this model the refinement phase replaces the more typical maintenance phase, during which network parameterisation can be further refined (Monti and Carenini, 2000). Alternatively the scope of the system may be further extended

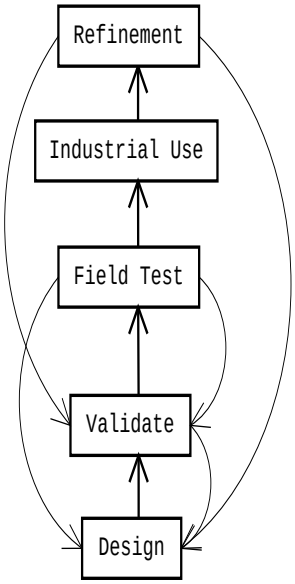

Figure 4: KEBN Lifecycle

(Laskey and Mahoney, 2000). In both cases the further development occurs as a refinement of the existing system rather than as a complete system rewrite.

In addition to changes to the network, the refinement process may improve the domain experts understanding of the problem. The refinement process may need to occur over an extended period of time. There may be a delay in obtaining sufficient new data and as Monti and Carenini (2001), discovered, experts do not adapt well to sudden change. The knowledge that their prior beliefs and reasoning are inconsistent with themselves may be poorly recieved.

KEBN is explained in more detail below, and then compared to a number of other development life cycles.

4.1 KEBN in detail

KEBN is specifically designed for the needs of Bayesian net construction. Its strength when

applied to Bayesian nets may be a disadvantage for other development methods for reasoning under uncertainty. The tasks in KEBN are described below.

4.1.1 Design

In this first phase the structure of the network is defined. As discussed in the Knowledge Engineering section, this can be done using expert elicitation methods (Monti and Carenini, 2000; Wang and Druzdzel, 2000) or by running causal discovery algorithms such as CaMML (Wallace and Korb, 1999) on collected data.

4.1.2 Validation

This step involves sensitivity analysis and accuracy testing. Sensitivity analysis can be critical in many commercial settings. Precision of measurement comes at a cost. Accuracy is important.

4.1.3 Field Testing

Alpha and beta testing of the networks provide feedback and an opportunity to find and fix problems. They provide a chance to test the application against the real world before its advice is trusted. If changes fail the field test, they return to the validation testing stage and are "checked" disgaurded. When the network is first being built, a field test failure may result in the network being redesigned from scratch.

4.1.4 Industrial Use

The product is used in industry and statistics are collected to allow for further refinement. The net is not modified until a refinement phase is started. The refinement phase as shown in Figure 4 does not lead directly back to industrial use, and validation and field testing must take place for the updated net is used.

4.1.5 Refinement

The network may be updated and refined to better fit its past data and correct for sub-optimal decisions. Small changes to parameterisation may be carried out and regression testing conducted. The network structure may also be extended to cope with more general situations. A change to structure will result in a need to revalidate and field test. If the validation and field testing is successful, the new structure may be used for a while before entering another refinement phase.

4.2 KEBN and the Fountain

The KEBN software development life cycle (SDLC) bares some resemblance to the fountain SDLC as presented by Henderson-Sellers and Edwards (1990) and shown in Figure 5.

Both are processes that allow new domain knowledge and expertise to cause a re-evaluation or redesign. KEBN combines analysis, conceptual design, component design and coding into one larger design phase. The KEBN validation phase can be seen as a form of unit testing. Field testing can be seen as whole system testing.

The strongest resemblance can be seen at the top of the KEBN method and fountain model. Both program use and industrial use lead on to an evolution / refinement phase. Despite all this, there are some important differences.

Decision support with Bayesian Nets should not require extensive coding. The design phase, described later in this section, is a combination of problem analysis and the development of a conceptual model. The methods of developing

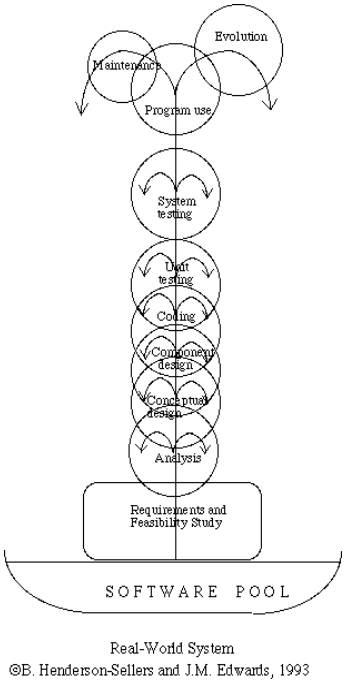

Figure 5: Fountain Lifecycle

this model are themselves a key part of KEBN and very different from the waterfall's conceptual and component design.

The models from the KEBN design phase may themselves be compiled. Validation can be a comparison against sample data and more often than not will lead to design changed. Field testing is the final fall back before a Bayesian Net based application is accepted for industrial use. A number of problems, such as over fitting, may be picked up at this stage. While both life cycles show a willingness to fall backwards, KEBN in practice makes more use of this and adopts a more prototype-like approach.

4.3 KEBN and the Prototype

KEBN differs from the classical prototyping approach and adds rigor to the flow between updating the prototype's ideas, implementing them and using them. This can be seen in KEBN's insistence that refinements flow back at least as far as validation rather than being immediately incorporated into a field test or industrial use. The classic prototyping methodology stops with the conversion to an operational system (as shown in Figure 6). KEBN acknowledges that as our knowledge of the state of the world changes, partly as a result of industrial use, the need for further refinement of the industrial model becomes more evident. This is particularly relevant for Bayesian nets in industrial applications given the mutual growth in understanding of both the net and the expert, as previously discussed. In KEBN unlike prototyping, there is no final conversion that ends the process of refinement.

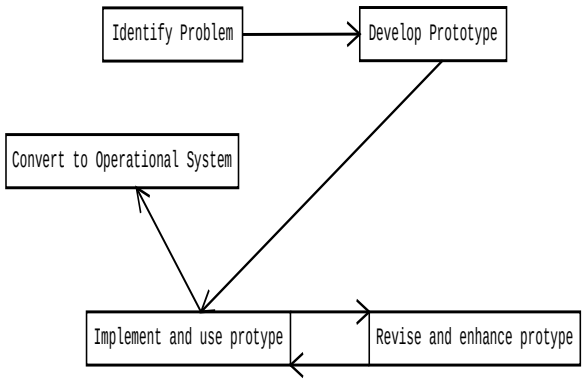

Figure 6: Prototyping Method

4.4 KEBN and the Spiral

KEBN shares the project initiation and conclusions phases of the spiral model, namely that a new iteration can start at any time when a hypothesis is put forward that the system's model could be improved. The KEBN cycle, by dropping back to the evaluation phase, then reviews whether this change might indeed improve the system. This is similar to the spiral that is initiated with a hypothesis for system improvement and then "involves a test of this hypothesis"

(Boehm, 1988). The difference is a matter of degree.

KEBN is a local development process for a system, a set of characteristics that are slowly being modelled closer to the reality. The possible changes are very limited and must be restricted to changes to the network structure or parameterisation. The flows back to design show a redevelopment of the network from scratch, most likely due to a change in the world rather than to an incremental improvement in knowledge.

The risk mitigation phase that looks at various possible technology approaches and implementation methods (the key element of the spiral) may in KEBN be non existent after the initial decision to use a Bayesian net is made. While a change to network structure may either improve the system or not, the validation and later field-testing of the change are now critical and must be extensively tested. In their paper "Network Engineering for Agile Belief Network Models" Laskey and Mahoney note the importance of "iterative refinement and enhancement of a prototype model" (Laskey and Mahoney, 2000). They propose starting with the core system and then expanding it to cope with more variables and less stringent assumptions as it is refined. While they praise the spiral method for making this possible, the internals of the spiral are largely ignored. They assess risk only at the start and in terms of which parts of the network may be expand or refined.

There is no other opportunity for risk assessment in the case of extending a Bayesian model. The method of expansion and technology is predetermined. It is this method and technology application that usually carries the varying degrees of risk for the different possibly implementations considered in a spiral development life cycle. Laskey and Mahoney come up with a system improvement, a hypothesis and then adapting the system to take account of it (by adding more nodes) before attempting to validate it. This approach fits much better to KEBN than to the spiral model, however at the meta level (that is between spirals) the two approaches may look the same.

5 Conclusions

In order for Bayesian nets to be better accepted and more widely used in industry, a methodology for developing large scale Bayesian nets is needed. Knowledge Engineering should focus on expert solicitation, Bayesian net learners, and the potentially positive interaction between them. The whole must be developed into a framework which allows both growth of the networks understanding and growth of domain experts understanding. This can be achieved by continual refinement of the system's structure and parameterisation. Increased understanding by experts and changes to the data that influences Bayesian net learners can fuel such refinement. The KEBN methodology along with various related ideas about capturing data for parameterisation and network abstraction may provide the start of such a solution.

6 Acknowledgments

We thank Kevin Korb for critiquing drafts of this manuscript and acknowledge his original KEBN design which formed the basis of this new adapted KEBN design.

References

Boehm, B. (1988). A spiral model of software development and enhancement, *IEEE Computer* **21**: 61–72.

Charniak, E. (1991). Bayesian networks without tears, *AI Magazine* **12**: 50–63.

CoIL (2000). Lessons about self-learning, Website of the Computational Intelligence and Learning Cluster network of excellence. Web page updated 17 of October, 2000. *http://www.dcs.napier.ac.uk/coil/news/feature49.html

Davidson, L. (1997). *Knowledge Extraction Technology for Terminology*, Masters, School of Translation and Interpretation, University of Ottawa.

DeMarco, T. (1979). *Structured Analysis and Systems Specification*, Prentice-Hall.

Druzdel, M. and Diez, F. (2000). Criteria for combining kowledge from different sources in probabalistic models, *Working notes for the workshop 'Fusion of Domain Knowledge with Data for Decision Support' 16th Annual conference on Uncertainty in Artificial Intelligence*, Stanford, CA, pp. 23–29.

Druzdzel, M. and Van der Gaag, L. (2000). Building probabilistic networks: "where do the numbers come from?" guest editors' introduction, *Knowledge and Data Engineering, IEEE Transactions on* **12**: 481–486.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, *I.E.E.E. Transactions Pattern Analysis and Machine Intelligence* pp. 721–741.

Haddawy, P. (1999). An overview of some recent developments in bayesian problem-solving techniques, *Artificial Intelligence Magazine* **20**: 11–19.

Heckerman, D. (1991). *Probabilistic Similarity Networks*, MIT Press, Cambridge, MA, USA.

Heckerman, D., Horvitz, E., and Nathwani, B. (1992). Towards normative expert systems: Part i, the pathfinder project, *Methods of Information in Medicine* pp. 90–105.

Henderson-Sellers, B. and Edwards, J. (1990). The object-oriented systems life cycle, *Communication of ACM* **33**: 71–79.

Jensen, F. (1996). *An Introduction to Bayesian Networks*, New York: Springer Verlag.

Keeney, R. and Raiffa, H. (1976). *Decisions with multiple objectives*, Wiley, NY.

Kennett, R., Korb, K. and Nicholson, A. (2001). Seabreeze prediction using bayesian networks, *Proc. of the 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Hong Kong, pp. 148–153.

Korb, K. (2002). Kebn, Website of CSE459. Web page viewed 26 October 2002. *http://www.csse.monash.edu.au/ korb/subjects/cse459/Lectures/L3/L3-4.ps.gz

Lam, W. and Segre, A. (2002). A distributed learning algorithm for bayesian inference networks, *Knowledge and Data Engineering, IEEE Transactions on* **14**: 93–105.

Laskey, K. and Mahoney, S. (2000). Network engineering for agile belief network models, *Knowledge and Data Engineering, IEEE Transactions on* **12**: 487–498.

Monti, S. and Carenini, G. (2000). Dealing with the expert inconsistency in probability elicitation, *Knowledge and Data Engineering, IEEE Transactions on* **12**: 499–508.

Musman, S., Chang, L. and Booker, L. (1993). Application of a real-time control strategy for bayesian belief networks to ship classification, *International Journal for Pattern Recognition and Artificial Intelligence* **7**: 513–526.

Musman, S. and Plehner, P. (n.d.). Real-time scheduling under uncertainty for ship self defense. submitted, available at `http://imsidc.com/ musman/ personal/RT-Sched.ps`.

Nikovski, D. (2000). Constructing bayesian networks for medical diagnosis from incomplete and partially correct statistics, *Knowledge and Data Engineering, IEEE Transactions on* **12**: 509–516.

O'Donnell, R. (2001). *Adaptation in Bayesian Networks*, Honours, School of Computer Science and Software Engineering, Monash University.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

Raiffa, H. (1968). *Decision Analysis*, Addison-Wesley.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence a Modern Approach*, Prentice Hall.

Wallace, C. S. and Korb, K. B. (1999). Learning linear causal models by MML sampling, *in* A. Gammerman (ed.), *Causal Models and Intelligent Data Management*, Springer-Verlag.

Wang, H. and Druzdzel, M. J. (2000). User interface tools for navigation in conditional probability tables and elicitation of probabilities in bayesian networks, *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Vol. 14, Morgan Kaufmann Publishers, Los Angeles, pp. 617–625. *http://www.pitt.edu/ druzdzel/ abstracts/uai00c.html