computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

REASONING UNDER UNCERTAINTY WITH BAYESIAN BELIEF NETWORKS ENHANCED WITH ROUGH SETS

Andrew J. Kornecki [1], Slawomir T. Wierzchon [2], Janusz Zalewski [3]

[1] Dept. of Electrical, Computer, Software and System Engineering, Embry Riddle Aeronautical University
Daytona Beach, FL 32114, USA
kornecka@erau.edu, http://faculty.erau.edu/korn/
[2] Faculty of Mathematics, Physics and Informatics, University of Gdańsk
Wita Stwosza 57, 80-952 Gdańsk-Oliwa, Poland
stw@ipipan.waw.pl, http://www.ipipan.waw.pl/~stw/
[3] Dept. of Software Engineering, Florida Gulf Coast University
Fort Myers, FL 33965, USA
zalewski@fgcu.edu, http://www.fgcu.edu/zalewski/

Abstract: *The objective of this paper is to present a new approach to reasoning under uncertainty, based on the use of Bayesian belief networks (BBN's) enhanced with rough sets. The role of rough sets is to provide additional reasoning to assist a BBN in the inference process, in cases of missing data or difficulties with assessing the values of related probabilities. The basic concepts of both theories, BBN's and rough sets, are briefly introduced, with examples showing how they have been traditionally used to reason under uncertainty. Two case studies from the authors' own research are discussed: one based on the evaluation of software tool quality for use in real-time safety-critical applications, and another based on assisting the decision maker in taking the right course of action, in real time, in the naval military exercise. The use of corresponding public domain software packages based on BBN's and rough sets is outlined, and their application for real-time reasoning in processes under uncertainty is presented.*

Keywords: *Bayesian Belief Networks, Rough Sets, Decision Uncertainty, Soft Computing.*

1. INTRODUCTION

Bayesian Belief Networks (BBN's) have been widely used in Industrial Information Systems for solving all types of computational problems with insufficient information and uncertainty [1,2]. This includes applications such as: water contamination [3], fault detection in an industrial process [4], fog forecasting at the airports [5], predicting software defects [6], inferring certification metrics of software [7], predicting hospital admissions for emergency [8], multisensor fusion for landmine detection [9], evaluation of risk in software development [10], modeling an air traffic control [11], cell signaling pathway modeling [12], reliability estimation [13], safety assessment [14] and risk evaluation [15] in computer-based systems, to name only a few from a long list. They have been also studied theoretically by a number of researchers, for example [16-17].

Although, in general, BBN's have been very effective, because they allow reasoning and making predictions based on small sets of probabilities with backwards inference, they are still based on probability theory. A significant disadvantage of BBN's is that, in realistic cases, they require extensive computations of the conditional probability values. In most of these studies, it has been recognized that this is one of the method's major limitations. Another disadvantage of BBN's is that they become less effective in case of missing probability values.

With this in mind, one wants to look at a complementary method of evaluating data in the input data set, which would not rely strictly on probability densities and could deal with missing values. One of the theories that offer such an approach, with values of data attributes and events measured by likelihoods rather than probabilities, is the rough sets theory [18-19].

Rough sets have been used since the early eighties [19], in a wide range of applications to reason about uncertainty, including data mining[20], medical diagnosis [21], robotic systems [22], decision making in medicine [23], cost estimation [24], modeling software processes [25-26], safety analysis [27], controller design [28], quality analysis

[29], fault diagnosis [30] and many others, for example prognostics [31].

The objective of this paper is to look at the combination of using BBN's and rough sets in decision making under uncertainty, and suggest the enhancement of pure Bayesian reasoning by additional use of rough sets for preliminary evaluation of data. The paper is structured as follows. The next two sections describe briefly basic concepts of Bayesian belief networks and rough sets, respectively. Then, in a separate section, several examples and two case studies are presented, giving an overview of the method developed for combining BBNs and rough sets for real-time computations. The final sections present general conclusions and suggestions for future work.

2. BAYESIAN BELIEF NETWORKS

The following section describes Bayesian Belief Networks from an application point of view rather than the underlying mathematics and statistics. Rev. Thomas Bayes developed this method of updating probabilities based on new information in the 1760s. It has been widely applied in probability and statistics for over 250 years.

"Essay Towards Solving a Problem in the Doctrine of Chances" was published after Bayes' death in 1763 [32]. It is the basis for the popular inversion formula for belief updating from evidence (E) about a hypothesis (H) using probability measurements of the prior truth of the statement updated by posterior evidence

$$P(H|E) = (P(E|H) * P(H)) / P(E)$$

where H is the hypothesis, E is the evidence, and P(x|y) is the conditional probability of x given y.

It is derived by the use of the joint probability definition

$$P(x, y) = P(x|y) * P(y) = P(y|x) * P(x)$$

that is then arranged as

$$P(x|y) = P(y|x) * P(x) / P(y)$$

where x = H and y = E.

Suppose we know from historical medical records that meningitis causes a stiff neck in 1 of 2 patients. We also know that 1 in 50,000 people have meningitis and that 1 in 20 people have a stiff neck. If you wake up with a stiff neck what is the probability that you have meningitis? How do you estimate it?

The hypothesis is that you have meningitis and the evidence is your stiff neck. Applying the Bayes formula yields:

$P(E|H) = 1 / 2 = 0.5$ or probability of a stiff neck when you have meningitis

$P(H) = 1 / 50,000 = 0.00002$ or probability of meningitis in the population;

$P(E) = 1/ 20 = 0.05$ or probability of a stiff neck in the population;

which yields in turn:

$$P(H|E) = (P(E|H) * P(H)) / P(E) = (0.5 * 0.00002) / 0.05 = 0.0002$$

which is the probability of having meningitis when you have a stiff neck. Much more complex models with multiple hypotheses and evidence sources can be constructed usually in a graph form relating cause to effect.

These belief networks are more recent concepts credited to Professor Judea Pearl with his construction and solution algorithms along with the work of many others [33].

A Bayesian belief network is a form of probabilistic graphical model. The belief network represents the joint probability distribution of a set of random variables with explicit independence assumptions described by a directed graph. In this research a Bayesian network is defined by a directed acyclic graph of nodes representing variables and arcs representing probabilistic dependency relations among the variables.

If there is an arc from node A to another node B, then variable B depends directly on variable A and A is called a parent of B. If the variable represented by a node has a known value then the node is said to be observed as an evidence node. A node can represent any kind of variable, be it a measured parameter, a latent variable or a hypothesis. Nodes are not restricted to representing random variables; this is what is "Bayesian" about a belief network.

In the following, an example is presented of three node networks that are structured as linear, converging, and diverging (Figure 1), with the use of Netica software program [34]. A different software package for Bayesian belief networks, named Hugin [35], is equally effective and simple to use.

The above examples are causal Bayesian networks where the directed arcs of the graph are interpreted as representing causal relations in some real domain with prior information. The directed arcs do not have to be interpreted as representing causal relations; however in practice knowledge about causal relations is very often used as a guide in drawing Bayesian network graphs, thus resulting in cause and effect Bayesian belief networks.

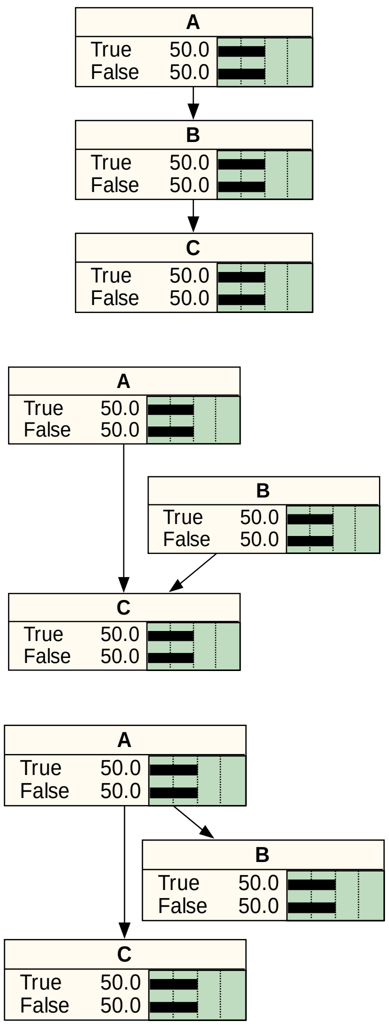


Fig. 1 – Examples of three-node Bayesian networks (top to bottom): linear, converging and diverging.

In the linear case on the top of Figure 2, A causes B that causes C, while in the converging model in the center, A is conditionally independent of B and both cause C, while in the diverging model at the bottom, A causes both B and C. In each case an effect is observed at node C illustrating the update of the joint probabilities when new information is incorporated into the network.

In the simplest case, a Bayesian network is specified by an expert and is then used to perform inference after some of the nodes are fixed to observed values. In order to fully specify the Bayesian network and fully represent the joint probability distribution, it is necessary to further specify for each node X the probability distribution for X conditional upon X's parents. The distribution of X conditional upon its parents may have many forms.

The following data (Table 1) are the conditional probabilities for the previous linear A, B, and C node network example (the top one) from Figure 2, where we observe that if C is true then column B is 0.8 true and 0.2 false. Prior to any observation as shown in Figure 1 the symmetry of the conditional probabilities makes the probability of true and false states equal to 0.5 for all nodes.

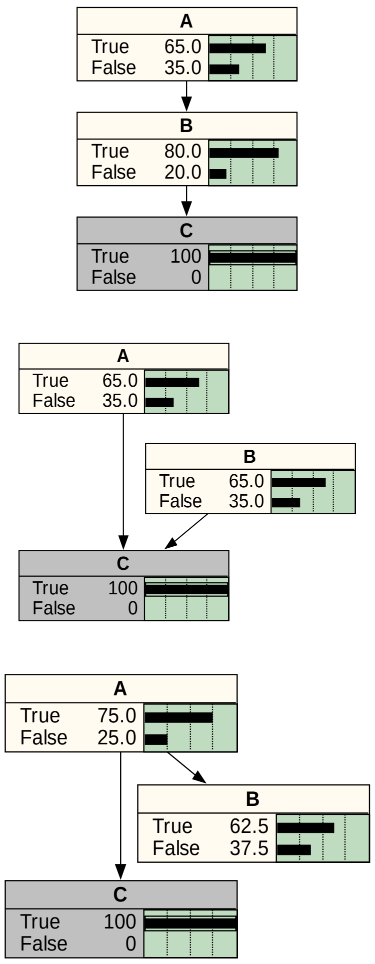


Fig. 2 – Causal relationships represented in a Bayesian belief network.

Table 1. Conditional probabilities for Figure 2

Node A		
True	False	
0.5	0.5	
Node B		
True	False	A
0.75	0.25	True
0.25	0.75	False
Node C		
True	False	B
0.80	0.20	True
0.20	0.80	False

The goal of inference is typically to find the distribution of a subset of the variables, conditional upon some other subset of variables with known values called the evidence or observations, with any remaining variables integrated out. This is known as the posterior distribution of the subset of the variables given the evidence. The posterior gives us a universal sufficient statistic for detection applications, when one wants to choose values for the variable subset which minimize some expected loss function, for instance the probability of decision error.

An example of inference in the center converging example from Figure 2 is to specify A as observed true and estimate the inferred values for B and C, with C updated by the new information but B unchanged since it is conditionally independent of A.

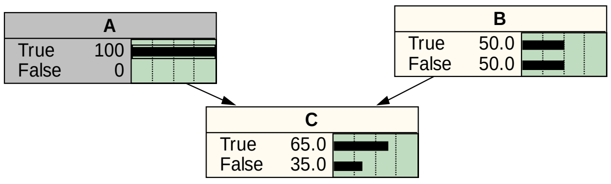


Fig. 3 – Effect of introducing evidence into a converging node in a Bayesian network.

In the divergence example from bottom of Figure 2, if A is observed then it infers new probability states for B and C that are dependent on A.

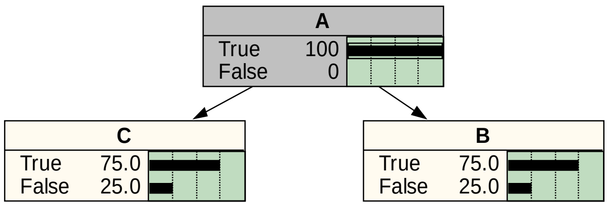


Fig. 4 – Effect of introducing evidence into a diverging node in a Bayesian network.

Questions about the dependence among variables can be answered by studying the graph alone. It can be shown that the conditional independence is represented in the graph by the graphical property of d-separation: nodes A and B are d-separated in the converging graph, given specified evidence nodes.

For belief reasoning a typical network is organized into three layers. The top layer is the causal variables, the middle layer is the reasoning variables, and the bottom layer is the effects variables. Four general classes of reasoning are defined for this three layer architecture. As an example the following five node network with three layers using binary random variables is used to illustrate the four principal reasoning strategies used in belief networks. The example network prior distribution has an equal probability for each variable state and is symmetric to illustrate the various conditional computations.

Diagnostic reasoning, illustrated in Figure 5, observes the effects of evidence and updates the middle reasoning variables and the top layer causal variables as shown in the example. This reasoning process diagnoses from an effect E of True to the cause B or in medical terms it is reasoning from symptom to disease. It also adjusts the probabilities for the middle reasoning layer C and the causal variable A and effect variable D.

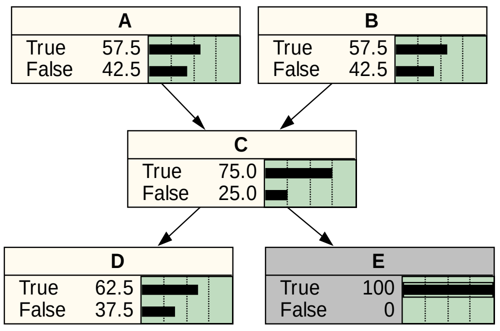


Fig. 5 – Illustration of diagnostic reasoning.

Predictive reasoning, illustrated in Figure 6, observes causal evidence and updates the middle reasoning variables and bottom layer effects variables as shown in the example. This reasoning process predicts from cause B to the effects D and E such as a patient saying that he is a smoker may focus on a certain set of symptoms. It also adjusts the probabilities for the middle reasoning layer C but not the independent causal variable A.

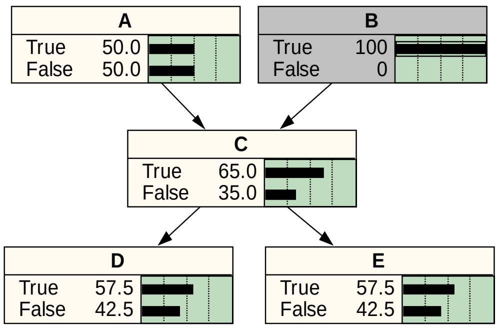


Fig. 6 – Illustration of predictive reasoning.

Intercausal or explaining reasoning, illustrated in Figure 7, observes both causal evidence and the middle layer reasoning evidence to update other causal variables as shown in the example. This reasoning process on C explains the mutual known cause B with unknown cause A and the common effects D and E. It is often interpreted as performing an experiment for explaining away cause A.

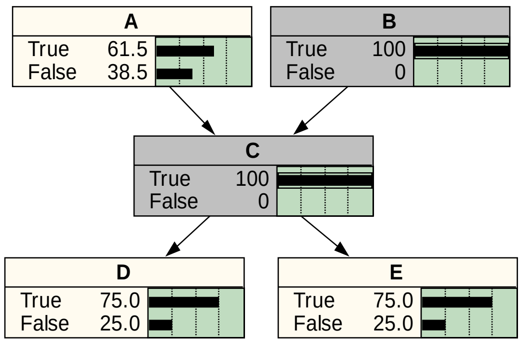


Fig. 7 – Illustration of intercausal (explaining) reasoning.

Combined reasoning, shown in Figure 8, observes causal evidence and effects evidence to update the middle reasoning variables as shown in the example. This reasoning process combines cause B and effect E to investigate the network conditional structure for reasoning variable C and the other cause A and the other effect D. This is a useful reasoning test for building and validating complex belief networks based on limited data and expert knowledge.

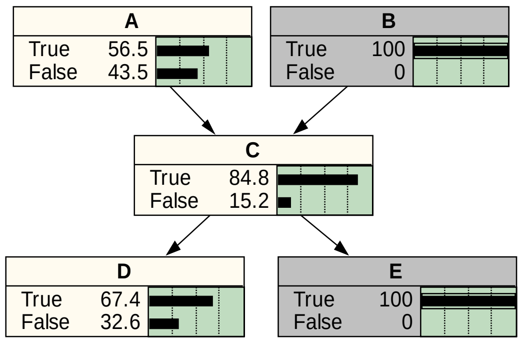


Fig. 8 – Illustration of combined reasoning.

3. ROUGH SETS

Rough Set Theory was invented by Zdzisław Pawlak to cope with limited perception of the surrounding world. The theory is especially helpful in dealing with vagueness and uncertainty in decision situations. Its main purpose is the "automated transformation of data into knowledge" [18]. The data are perceived in terms of objects and their features, i.e., values of the attributes used to characterize these objects. The knowledge deduced from these data is expressed in terms of *surely* and *possibly* statements describing notions of interests. More formally, such descriptions can be divided into so-called lower and upper approximations of entire notions. In the rest of this section, we describe a qualitative procedure containing all steps needed to form appropriate description of the concepts under considerations.

3.1 EXPLANATION OF A NOTION OF A ROUGH SET

First, let us illustrate intuitively a concept of a rough set, comparing it to an ordinary set and a fuzzy set, in a single dimension. Figure 9 shows such an intuitive illustration. For an *ordinary set*, the interval [*A*,*B*] in Figure 9, all its elements, that is, real numbers from this interval (assuming *x* represents a real axis), have values of their membership function equal to 1.0.

For a *fuzzy set*, elements on the set boundaries, that is, in the intervals [*A*,*C*] and [*D*,*B*], have values of their membership function equal to a fraction, a number from the interval [*0.0, 1.0*]. This means that these elements only partially belong to the set, to the extent specified by the value of a membership function.

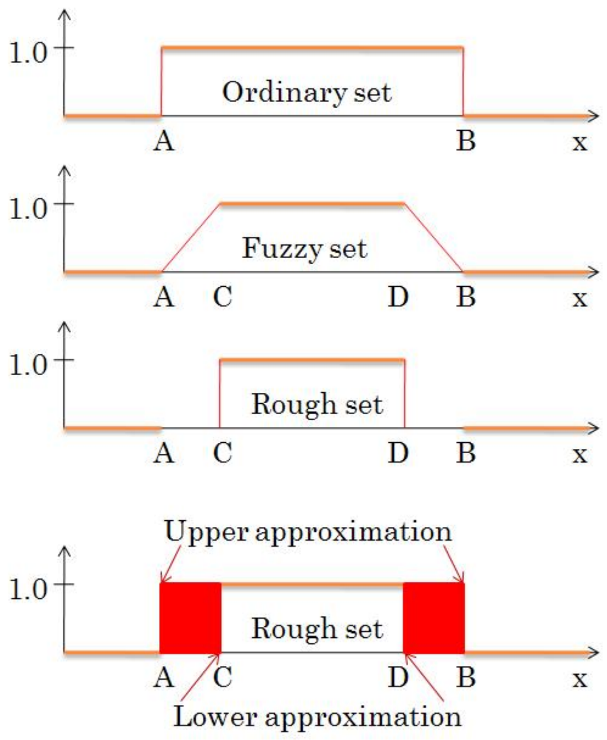


Fig. 9 – Intuitive illustration of a rough set vs an ordinary and a fuzzy set.

In contrast to the traditional concepts of a set, whether ordinary or fuzzy, for a rough set one cannot determine, even partially, the membership of the elements on the set boundary. Therefore, the value of the membership function for boundary elements of a set is undetermined. A rough set can only be described by its approximations, as illustrated in the lower part of Figure 9.

To express these intuitive concepts a bit more formally, we start from a relational database, i.e., a table with rows corresponding to objects and columns corresponding to the attributes. Each entry of the table represents attribute value of a corresponding object (i.e., its feature). In rough set formalism the database is considered as an information system, i.e., a quadruple

$IS = (U, A, V, f)$,

where $U = \{u_1, \ldots, u_n\}$ stands for a (usually finite) set of objects,

$A = \{a_1, \ldots, a_m\}$ *is a set of attributes*,

$V = V_1 \cup \ldots \cup V_m$,

where V_i is the domain of i-th attribute, and

$f: U \times A \to V$

is a so-called information function providing the description of objects, i.e., $f(u_i, a_j)$ assigns a value of j-th attribute to i-th object. The above mentioned concepts are illustrated in Table 2, in which

$U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$,

$A = \{a_1, a_2, a_3\}$, and

$V = V_1 \cup V_2 \cup V_3, = \{\text{Low, Med, High}\}$
$\cup \{\text{Min, Under, Over, Max}\} \cup \{\text{yes, no}\}$.

Usually the set A is decomposed into two disjoint subsets $A = C \cup D$ and the attributes from C are used to characterize objects and form so-called *condition* attributes, while the attributes in D are so-called *decision* attributes and they are used in decision-making or classification tasks. An information system with specified condition and decision attributes is called *decision table*. For the example in Table 2, attributes a_1 and a_2 could be interpreted as condition attributes, that is, certain parameters of an object, with values from the range {Low, Med, High} and {Min, Under, Over, Max}, respectively, and attribute a_3 – as a decision attribute, with values "yes" and "no." Hence, Table 2 can be viewed as a decision table.

Table 2. Example of an information system

$f: U \times A \to V$	**Attributes A**		
Objects U	a_1	a_2	a_3
u_1	Low	Max	yes
u_2	Low	Min	no
u_3	Med	Under	no
u_4	Med	Under	yes
u_5	High	Over	no
u_6	Low	Over	yes
u_7	High	Over	no
u_8	Low	Min	no

Because of the limited knowledge, we cannot fully discern objects, i.e., there are such objects u, v in U that $f(u, c) = f(v, c)$ for all the condition attributes c. This fact leads to the notion of *indiscernibility* relation E being in fact an equivalence relation on U. For example, for the information system in Table 2, objects u_2 and u_8 are indiscernible. So are objects u_5 and u_7.

It appears that in many cases we can identify proper subsets C' of C such that the indiscernibility relation $\mathsf{E}_{C'}$ induced by the attributes in C' is identical with the original relation E. Such sets of attributes are called *reducts*. Existence of reducts proves that not all of the attributes are necessary to form the equivalence classes. In other words identifying reducts allows more economic description of objects as we need smaller number of descriptors (features) to characterize these objects. Unfortunately, from a computational point of view this is an NP-hard task. No such reducts exist for the example shown in Table 2.

3.2 DEFINITION OF A ROUGH SET

Now we are ready to introduce the key concepts of rough set theory. Let B be a subset of the condition attributes and let $[v]_B$ stand for an equivalence class, i.e., a set of objects u in U with identical description (narrowed to the set B) as the object v. The subset X of U can be characterized using information in B by means of so-called *B-lower* and *B-upper* approximations defined as:

$$B(X)_* = \{u \in U \mid [u]_B \subseteq X\} \qquad (1a)$$
$$B(X)^* = \{u \in U \mid [u]_B \cap X \neq \emptyset\} \qquad (1b)$$

The lower approximation of X is the collection of objects which can be viewed *surely* as members of the set X, while the upper approximation of X is the collection of objects that *possibly* are members of X. Obviously $B(X)_* \subseteq B(X)^*$. If $B(X)_* = B(X)^*$ we say that X is B-definable and otherwise it is only partially definable. The set $BN_B = B(X)^* - B(X)_*$ is called a *B-boundary* region; it specifies the objects that cannot be classified with certainty to be either inside X, or outside X.

There are many grades of partial definability. We say that the set X is:

- roughly B-definable
  iff $B(X)_* \neq \emptyset$ and $B(X)^* \neq U$,
- internally B-indefinable
  iff $B(X)_* = \emptyset$, $B(X)^* \neq U$,
- externally B-indefinable
  iff $B(X)_* \neq \emptyset$, $B(X)^* = U$,
- totally B-indefinable
  iff $B(X)_* = \emptyset$, $B(X)^* = U$.

Obviously, if $B = C$, i.e., the full set of condition attributes is used, we omit the prefix B- in all above definitions. In such a case, a set X is characterized by the pair (X_*, X^*) and we say that X is a rough set (or B-rough set).

To illustrate these newly introduced concepts for the information system in Table 2, let's distinguish between condition attributes a_1 and a_2, and a decision attribute a_3. Values of a_1 and a_2, can be interpreted as vague measurements (evaluations) of certain parameters of a technical system, and a_3 can be viewed as a decision (control) based on these evaluations. Let the set B be the entire set of condition attributes, $C = B = \{a_1, a_2\}$. If the equivalence class $[v]_B$ is defined as

$$[v]_B = \{ \{u_1\}, \{u_2, u_8\}, \{u_3, u_4\}, \{u_5, u_7\}, \{u_6\} \}$$

then the set $X = \{ u \mid a_3(x) = \text{yes} \}$ has the following approximations:

$$B(X)_* = \{u_1, u_6\}$$
$$B(X)^* = \{u_1, u_3, u_4, u_6\}$$

This determination can be also illustrated in the table representing the information system under consideration (Table 3).

Table 3. Illustration of lower and upper approximations for a sample information system

f: $U \times A \to V$	Condition attributes C		Decision attribute D
Objects U	a_1	a_2	a_3
u_1	Low	Max	yes
u_2	Low	Min	no
u_3	Med	Under	no
u_4	Med	Under	yes
u_5	High	Under	no
u_6	Low	Over	yes
u_7	High	Over	no
u_8	Low	Min	no

The set X for a decision variable's value equal "*yes*" has three corresponding objects, u_1, u_4 and u_6. Given values of the specific condition attributes $B = \{a_1, a_2\}$, two of these objects, u_1 and u_6, lead *surely* to this decision value (*yes*). Thus, u_1 and u_6, form the lower approximation of set X. This is illustrated with heavy shading in Table 3. If, however, we take the third object, u_4, the values of its condition attributes, $\{ a_1=\text{Med}, a_2=\text{Under}\}$, can produce two different values of the decision attribute: "*yes*" for object u_4, and "*no*" for object u_3. Thus, objects with these values of the condition attributes belong *possibly* to the set X, which is illustrated with light shading in Table 3. Obviously, objects in the non-shaded lines do not belong to X.

To get a numerical characterization of the "roughness" of a set X we introduce so-called *accuracy of approximation*

$$\alpha_{B(X)} = |B(X)_*| / |B(X)^*| \qquad (2)$$

where the symbol $|Y|$ stands for the cardinality of the set Y. X is said to be crisp (or precise) with respect to the set of attributes B if and only if $\alpha_{B(X)} = 1$, and otherwise X is said to be rough (or vague) with respect to B.

Another characterization of the set of objects can be obtained by introducing so-called *rough membership* function $\mu_{B,X} : U \to [0,1]$ defined as follows

$$\mu_{B,X}(u) = |[u]_B \cap X| / |[u]_B| \qquad (3)$$

With such a definition a relationship between rough and fuzzy sets theory is established. More particularly, $\mu_{B,X}(u)$ determines the degree in which object u described by the set B of attributes belongs to the concept (equivalence class) X. Further, we can relax the definitions of the lower and upper approximation, namely

$$B_\beta(X)_* = \{u \in U | \mu_{B,X}(x) \geq \beta\} \qquad (4a)$$
$$B_\beta(X)^* = \{u \in U | \mu_{B,X}(x) > 1\text{-}\beta\} \qquad (4b)$$

where $0 \leq \beta \leq 1$. If $\beta = 1$ we obtain original definitions (1a) and (1b).

3.3 ROUGH RULES

In practical applications of interest are the sets of objects with identical set of decision attributes, that is, we define X as the set of objects satisfying the equality $f(x_1, d) = f(x_2, d)$ for all attributes d in D. If D is, for example, a set of diseases then X is a set of persons suffering on a particular disease, and the equivalence classes $[x]_B$ contain patients with identical symptoms (restricted to the set B). Hence, it is natural to find such condition attributes which can be used to discriminate between different diseases. This leads us to the practical aspects of rough set theory: rough rules.

More formally, given an information system IS = (U, A, V, f) and a subset $B \subseteq A$ we start from the set of atomic formulae, called also descriptors, being expressions of the form $a = v$, where $a \in B$ and $v \in V_a$. Next, we define the set of all possible formulae $F(B,V)$ containing all atomic formulae and being closed with respect to the logical connectives: $\neg$ (negation), $\wedge$ (conjunction) and $\vee$ (disjunction).

If φ is an atomic formula of the form $a = v$, then its meaning (semantics) is as follows:

$$||\varphi|| = \{u \in U \mid f(u,a) = v\}$$

If φ is a compound formula then

$$||\neg\varphi|| = U \setminus ||\varphi||,$$
$$||\varphi\wedge\varphi'|| = ||\varphi|| \cap ||\varphi'||, \text{ and}$$
$$||\varphi\vee\varphi'|| = ||\varphi|| \cup ||\varphi'||.$$

Now a decision rule is any expression of the form

$$\varphi \Rightarrow (d = v),$$

where d is a decision attribute; the formula φ is said to be predecessor (or ancestor) of the rule and the formula ($d = v$) – its successor (or consequent). We say that the decision rule:

$$\varphi \Rightarrow (d = v)$$

is true in the information system IS, if

$$||\varphi|| \subseteq ||(d = v)|| \text{ and } ||\varphi|| \neq \varnothing.$$

Deeper classification of the rules is given in [36].

For instance the rule

$$r_1: (a_1 = \text{Low}) \wedge (a_2 = \text{Max}) \Rightarrow (a_3 = \text{yes})$$

is true in the information system from Table 3, while the rule

$$r_2: (a_1 = \text{Med}) \wedge (a_2 = \text{Under}) \Rightarrow (a_3 = \text{yes})$$

is only partly true because

$$||(a_1 = \text{Med}) \wedge (a_2 = \text{Under})|| = \{u_3, u_4\}$$
$$\text{and } ||(a_3 = \text{yes})|| = \{u_1, u_4, u_6\};$$

thus $||(a_1 = \text{Med}) \wedge (a_2 = \text{Under})|| \not\subset ||(a_3 = \text{yes})||$. Detailed remarks on inducing rules from information systems can be found, for example, in [37].

To characterize the rules numerically, a number of measures can be introduced; *support* and *confidence* are most popular. The former is defined as the number of objects satisfying both predecessor and successor, while the latter as the conditional probability that the consequent is satisfied provided the ancestor is satisfied. In case of rule r_2 its support, $sup(r_2) = 1$, and its confidence, $conf(r_2) = ½$.

The already mentioned process of "transformation of data into knowledge" translates now into refining the dependencies between sets of attributes. Intuitively, if C and D are two sets of attributes, we say that D depends totally on C, if all values of the attributes from D are uniquely determined by values of attributes from C. This is functional dependency known from database theory.

Rough set theory enables relaxing this definition by introducing a dependency in a degree $k \in (0, 1]$. An interested reader is referred to [19] and [38] for details. There are at least two successful computer programs allowing rough data analysis: Rosetta [39] downloadable from the following website: http://rosetta.lcb.uu.se/general/ and LERS [40].

Finally if a new object is introduced into the data set with the attribute value missing, one could attempt to determine this value by using the previously generated rules. This is explained in the next section.

3.4 HANDLING THE MISSING VALUE IN A ROUGH SET

Grzymala-Busse describes several algorithms of dealing with missing values in information systems, based on three types of such values [41]:

- those which are lost and no longer available
- totally irrelevant values, and
- partially relevant values.

They are marked in Table 4, using the following symbols: a question mark "?" for not available values, an asterisk "*" for irrelevant values, and a dash "-" for partially relevant values.

Table 4. Information system with some missing values

$f: U\times A \to V$	Condition attributes C		Decision attribute D
Objects U	a_1	a_2	a_3
u_1	?	Max	yes
u_2	Low	Min	no
u_3	Med	Under	no
u_4	-	Under	yes
u_5	High	Over	no
u_6	Low	Over	yes
u_7	High	Over	no
u_8	Low	*	no

To calculate the approximations, one has to start with the meaning of the atomic formulas in a given information system. For the information system in Table 2, these meanings, called also *blocks* in [41] are as follows:

$$||a_1 = \text{Low}|| = \{ u_1, u_2, u_6, u_8 \}$$
$$||a_1 = \text{Med}|| = \{ u_3, u_4 \}$$
$$||a_1 = \text{High}|| = \{ u_5, u_7 \}$$
$$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$$
$$||a_2 = \text{Under}|| = \{ u_3, u_4 \}$$
$$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7 \}$$
$$||a_2 = \text{Max}|| = \{ u_1 \}$$

These sets have to be modified for an information system with missing values in Table 4, as follows. For the missing value of the attribute a_1, which is not available for object u_1 and marked "?", object u_1 has to be removed from all blocks created for this attribute, that is, block $||a_1 = \text{Low}||$ will change to:

$$||a_1 = \text{Low}|| = \{ u_2, u_6, u_8 \}$$

with two other blocks for a_1 remaining unchanged, because they do not include objects with lost value of a_1.

For the missing value of the attribute a_2, which is irrelevant and marked "*", its corresponding object, u_8, has to be included in blocks for all values of this attribute, which will lead to the following modifications:

$$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$$
$$||a_2 = \text{Under}|| = \{ u_3, u_4, u_8 \}$$
$$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7, u_8 \}$$
$$||a_2 = \text{Max}|| = \{ u_1, u_8 \}$$

Finally, for the missing value of the attribute a_1, which is marked "-", as partially relevant, respective object u_4 has to be added to the blocks containing objects corresponding to the decision attribute's value the same as the value of this decision attribute for the partially relevant value. In case of Table 4, the partially relevant value of attribute a_1 for object u_4 corresponds to the decision attribute's value "yes". Thus, this attribute's value is relevant to this particular decision attribute, and this is the meaning of the term "partially relevant". Two other objects exist, which have "yes" as their decision attribute's value: u_1, whose value of attribute a_1 is unavailable, so we drop it from consideration, and u_6, whose value of a_1 equals *Low*; therefore u_4 has to be added to the block, which contains a_1 = Low, because it is partially relevant to corresponding decision attribute.

So the final list of blocks looks as follows:

$||a_1 = \text{Low}|| = \{ u_2, u_4, u_6, u_8 \}$
$||a_1 = \text{Med}|| = \{ u_3, u_4 \}$
$||a_1 = \text{High}|| = \{ u_5, u_7 \}$
$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$
$||a_2 = \text{Under}|| = \{ u_3, u_4, u_8 \}$
$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7, u_8 \}$
$||a_2 = \text{Max}|| = \{ u_1, u_8 \}$

Because of the limited length of this paper, we can only mention here that for further computations the so called *characteristic sets* have to be calculated, for each object, which is done as follows:

1) The characteristic set ***K*** of an object is defined as an intersection of blocks for specific values of the attributes for this object.
2) If the value of an attribute is irrelevant "*" or unavailable "?", then the entire universe *U* is taken as a corresponding block for this attribute.
3) If the value of an attribute is partially relevant "-", then for this specific block it is substituted by a union of blocks representing particular values of the attributes for the corresponding decision attribute's value.

A more formal presentation of these concepts, with respective algorithms, is given in [41]. Below we present the computation of characteristic sets for the list of blocks corresponding to Table 4.

$$K_{u1} = U \cap \{ u_1, u_8 \} = \{ u_1, u_8 \}$$
$$K_{u2} = \{ u_2, u_4, u_6, u_8 \} \cap \{ u_2, u_8 \} = \{ u_2, u_8 \}$$
$$K_{u3} = \{ u_3, u_4 \} \cap \{ u_3, u_4, u_8 \} = \{ u_3, u_4 \}$$
$$K_{u4} = \{ u_2, u_4, u_6, u_8 \} \cap \{ u_3, u_4, u_8 \} = \{ u_4, u_8 \}$$
$$K_{u5} = \{ u_5, u_7 \} \cap \{ u_5, u_6, u_7, u_8 \} = \{ u_5, u_7 \}$$
$$K_{u6} = \{ u_2, u_4, u_6, u_8 \} \cap \{ u_5, u_6, u_7, u_8 \} = \{ u_6, u_8 \}$$
$$K_{u7} = \{ u_5, u_7 \} \cap \{ u_5, u_6, u_7, u_8 \} = \{ u_5, u_7 \}$$
$$K_{u8} = \{ u_2, u_4, u_6, u_8 \} \cap U = \{ u_2, u_4, u_6, u_8 \}$$

As explained in [41], computation of lower and upper approximations, depends on their definitions. The author presents three such definitions and for one of them:

$$B(X)_* = \{ u_1, u_4, u_6, u_8 \}$$
$$B(X)^* = \{ u_1, u_2, u_4, u_6, u_8 \}$$

The interpretation of this result is such that the missing values cause broadening of the potential span for the lower approximation, because they have to be inferred from the rest of the set. The upper approximation can change either way, because the missing values change the entire structure of a set.

4. COMBINATION OF BBN'S WITH ROUGH SETS

This section describes several examples and two case studies related to Bayesian networks and rough sets. First, we give a background on applying Bayesian networks to software quality evaluation. Next, we discuss a case study on the assessment and qualification of software tools for real-time safety-critical systems. Finally, we present a method for combining Bayesian networks and rough sets in decision making under uncertainty, and discuss the operation of two public domain tools, assisting in real-time decision making.

4.1 USE OF BBN'S FOR SOFTWARE QUALITY EVALUATION

In recent years, these authors have dealt with various aspects of assessing software quality in real-time safety-critical applications [42-44]. The basic idea to apply Bayesian networks in such problems comes from multiple previous attempts to assess various software properties in critical applications, which are briefly outlined below.

A. Application of BBN's to Assess Software Quality. In one of the first studies reported [45], the authors addressed the eternal question: "Can we predict the quality of our software *before* we can use it?", by applying BBN's to assess the *defect density* as a measure of software quality. A simplified diagram from their study is presented in Figure 10. The nodes were built based on the understanding of life-cycle processes, from requirements specification through testing.

The probabilities of respective states were based on the analysis of literature and common-sense assumptions about the relations between variables. The node variables are shown on histograms of the predictions obtained by execution of the network after the evidence entered (the evidence is represented by nodes with probabilities equal to 1.0). As the authors say, the advantage of their model is that it "provides a way of simulating different events and identifying optimum courses of action based on uncertain knowledge."

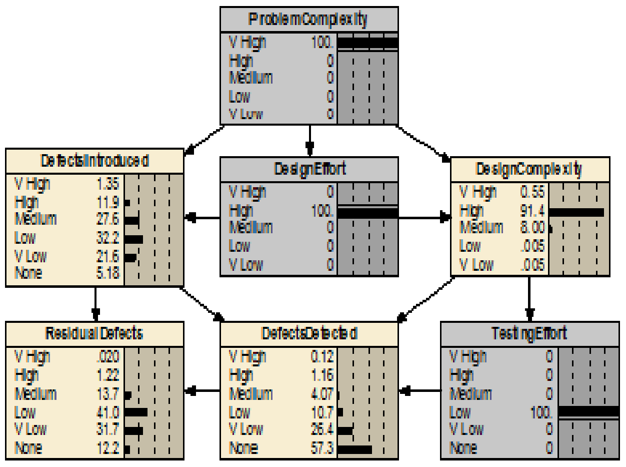


Fig. 10 – Simplified BBN for assessing software fault density [45].

B. Use of BBN's in Assessment of Software Safety. Gran et al. [46] applied BBN's to address safety assessment of software for acceptance purposes, in a more comprehensive way, using multiple information sources, such as complexity, testing, user experience, system quality, etc.

Their BBN network for system quality, which is only a part of the entire model, is shown in Figure 11. It involves two root nodes: *UserExperience* and *VendorQuality*, and a number of leaf nodes, corresponding to observable variables, of which *QualityMeasures* is of particular importance. This node shows evidence about the system quality, grouping quality attributes, such as readability, structuredness, etc., and can be expanded further.

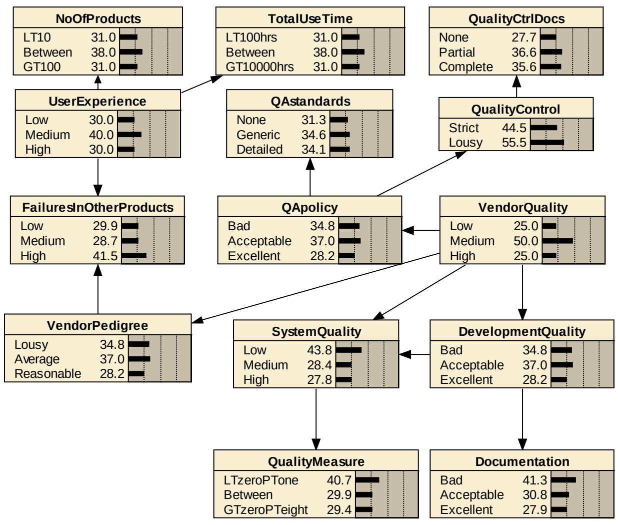


Fig. 11 – BBN for the system quality in safety assessment [46]

Other observable variables include *FailuresInOtherProducts*, those related to the user experience (*NoOfProducts* and *TotalUseTime*), as well as those related to quality assurance policy. When evidence becomes available, entering respective observation data into these nodes and executing the network provides assessment of the variable in question, which in this case is *SystemQuality*.

The authors note, however, that their example is intended more as an illustration of the method rather than as a real attempt to compute the quality of the system. Their probability assignments to the node variables were chosen somewhat *ad hoc*, and not based on any deeper analysis of the problem. However, as the authors say in conclusion, the results of the study were positive and showed "that the method reflects the way of an assessor's thinking during the assessment process."

C. BBN's in Dependability and Reliability Assessment. [47] used BBNs to formalize reasoning about software dependability to facilitate the software assessment process. They constructed a network for evaluating dependability of a software-based safety system. It used the data associated with two primary assumptions: the excellence in development (called a process argument) and failure-free statistical testing (called a product argument). The network topology includes taking into consideration variables such as: Test Failures, Operational Failures, Initial Faults, Faults Found, Faults Delivered, and System PFD (Probability of Failure per Demand). The probability distributions have been derived from a sample of programs from an academic experiment.

The authors were interested in estimating the probabilities of failure during acceptance testing and during the operational life of the product (represented by two variables mentioned in previous paragraph), given the prior probabilities and observed events. In particular, positive results of an acceptance test allowed deriving numerical estimates about the PFD and operational performance of the product.

Helminen [13] used BBN's to attack the problem of software reliability estimation. His primary motivation to apply BBN's was that they allow all possible evidence (large number of variables, different potential sources, etc.) to be used in the analysis of the reliability of a programmable safety-critical system. The essential characteristic of such systems is that they involve a significant number of variables related to reliability, with very limited evidence.

The reliability of such systems is modeled as a *probability of failure*, that is, the probability that the programmable system fails when it is required to operate correctly. To develop an estimate of probability of failure, the authors built a series of BBN models, using evidence from such sources, as the system development process, system design features, and pre-testing, before the system is deployed. This is later enhanced by data from testing and operational experience.

The essential part of this work was building BBN models for various operational profiles for multiple test cycles, involving continuous probability distributions. As a result, using BUGS software that combines Bayesian inference with Gibbs sampling, via Markov chain Monte Carlo (MCMC) simulation, it was possible to estimate, how many tests had to be run for a single system in a particular operational environment to achieve certain level of reliability. To decrease the huge number of necessary tests, multiple operational profiles for the same system were used, which required building replicated BBN models to include other profiles' evidence. In essence, by expanding the BBN models further, this approach also allows reliability estimation over the entire lifespan of the software product, but respective experiments have not been conducted in this study.

4.2 CASE STUDY IN SOFTWARE TOOL EVALUATION

To test the applicability of BBNs in software assessment, we applied this technique to evaluate the software development tools used in real-time safety critical applications in avionics. The data for the project were taken from experiments described in detail elsewhere [43,48]. The experiments involved applying a number of specific criteria, including: *efficiency* of the generated code, to conduct forward evaluation regarding the quality of code, and *traceability*, to allow backward evaluation regarding the tool capability of maintaining the right requirements. To evaluate the tool during its operation from perspective of the functions it provides and the ease of use, two additional criteria seemed to be appropriate: *functionality* and *usability*. The exact process of choosing the criteria is described in [48]. For criteria selected that way, a series of experiments were conducted, with six industry-strength tools applied to embedded software development. The above mentioned criteria were quantified using the following measures:

- *Efficiency* measured as code size (in LOC)
- *Usability* measured as development effort (in hours)
- *Functionality* measured via the questionnaire (on a 0-5 points scale)
- *Traceability* measured by manual tracking (in number of defects).

Data for some measures were collected in multiple aspects, for example, data involving the development effort were divided into four categories: preparation, modeling and code generation, measurements, and postmortem (including report writing). Details of the software requirements and actual experimental results are discussed in [43].

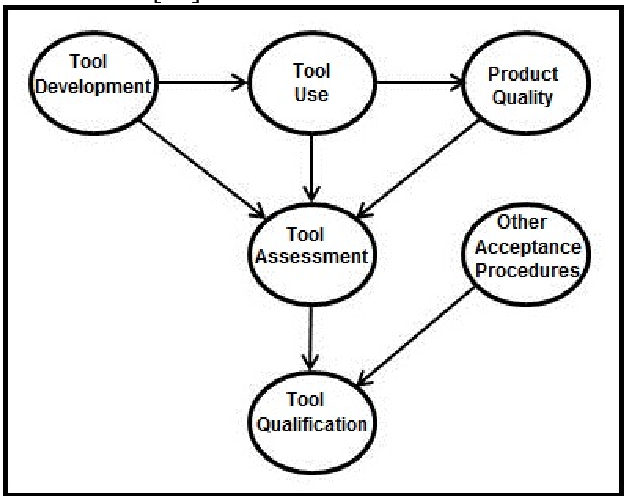


Fig. 12 – High-level BBN model for software tool evaluation.

Based on the adopted model of the tool evaluation process, and the results of experiments with the selected evaluation criteria outlined above, our high-level model of a BBN for tool assessment is illustrated in Figure 12. Its primary assumptions are that the tool assessment process should involve the following mutually interrelated factors:

a) development of the tool itself (including the process, vendor quality and reputation, their quality assessment procedures, etc.),

b) the tool use (including experimental evaluation based on predefined criteria, but also previous user experiences with this tool, etc.)

c) quality of the products developed with this tool, based on product execution, static code analysis, etc.

Based on the results of this analysis and other acceptance procedures (such as, legal aspects, independent experts opinions, etc.), the tool qualification process can be completed, as reflected in a BBN in Figure 13. Because of the limited data obtainable from experiments, we only deal with *ToolUse* part of the diagram in Figure 12. The logic of the BBN is similar to the ones reported in [14], where they had no real probability data, and [46], where the conditional probability values "were estimated based on judgments in a brainstorming activity among the project participants."

For the experimentally collected data for six tools, nicknamed L, M, N, O, P and Q, a sample tool assessment BBN is shown in Figure 13 for a tool, which is likely to pass the qualification process with 80% confidence at the level *MediumToHigh* or *High*.

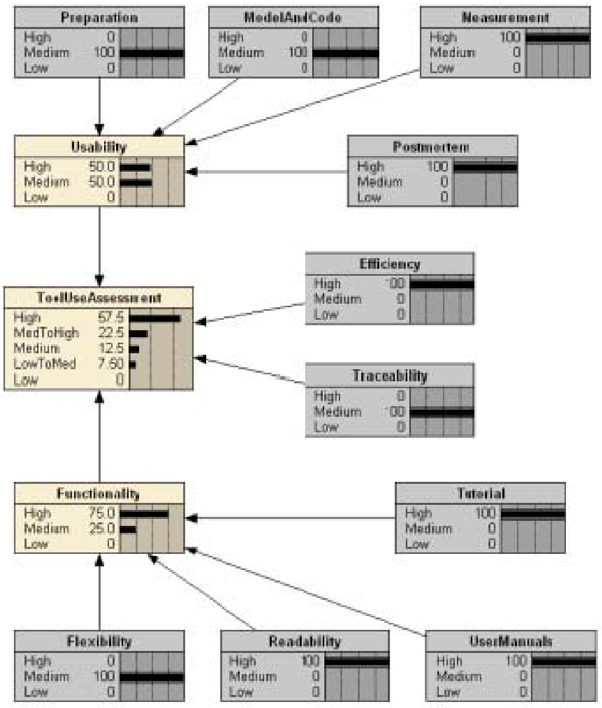


Fig. 13 – BBN to assess numerically quality of tool L.

4.3 REAL-TIME APPLICATION: THE AUSTRALIAN NAVY EXERCISE

As visible from previous examples, the principle of using a BBN for reasoning under uncertainty is that when the evidence about the state of one of the nodes (variables) becomes available, the rest of the network is also updated according to the conditional probability tables and dependency relations among the nodes. However, an updating process becomes a problem, if the new evidence is distorted or missing. This situation does not look that difficult in off-line computations, such as those discussed in subsections above, because one can do additional experiments and wait for the data when they will become complete. But if one wants to use BBN's for situation assessment in real time, when missing or distorted data come into play, as in circumstances such as sensor noise or sensor failure, especially over extended period of time, then the value of Bayesian reasoning may become problematic.

In general, this issue comes into play when there is no information on certain behavior or some information previously available becomes scarce or unavailable. Then using a rough set theory can help filling the gap caused by such circumstances. To illustrate this concept, we present a case study of the Australian Naval Exercise [49].

In this case study, there are two naval military forces called Blue and Orange that are hostile towards each other, and a country that the Orange forces obtain fuel supplies from and the Blue forces treat as neutral. The Blue forces have communications and surveillance facilities that the Orange forces want to destroy. Blue have set up a restricted area that contains the communication facilities and will consider any military activity or transportation of supplies hostile. Orange have a supply route that passes through the restricted area that it wants to defend.

Blue monitor the restricted area via sensors and reconnaissance. Orange vessels that are likely to be detected are Guided Missile Frigates (*FFG* in Figure 14), Free Mantle Class Patrol Boards (*FCPB*), and *Communication* vessels. Oil *Tankers* from the neutral country can also be detected. The position, mobility, and communications activity of the vessel are also recorded to try to determine the intent of the Orange forces.

The Bayesian network in Figure 14 is used to try to determine what the intentions of the Orange forces are and how to respond to it by entering the findings from the sensors and reconnaissance into the appropriate nodes. In essence, based on this information entered into the bottom nodes, the Bayesian network recalculates the variables in all other nodes, and the value of a variable in node *BlueCOA* makes a suggestion to the decision maker, what would be the most appropriate Course of Action (COA) at any given time.

The situation is more complicated when some of the sensor or reconnaissance data are missing, for example, due to a sensor failure or temporary or permanent unavailability of the reconnaissance. The BBN, which does the calculations, still expects receiving new data, because the command unit has to assess the situation and make respective decisions in real time. Even though the BBN can still operate, the missing data make its assessments less and less accurate when the time progresses.

In such case, we try to employ a rough set theory, particularly in its part dealing with the missing values. The essential idea is as follows. If we treat specific variables from the BBN network as attributes of the information system (rough set), with one of them being the decision attribute and all remaining ones – condition attributes, then we can determine (with some level of accuracy) the missing values of the attributes, using the reasoning presented briefly in the section on rough sets and described in more detail in [41]. In plain language, this would be equivalent to deriving the approximate value of a certain variable based on the context information. A sample of a respective information system is illustrated in Figure 15, for the Australian

Naval Exercise, using a rough set tool Rosetta [39].

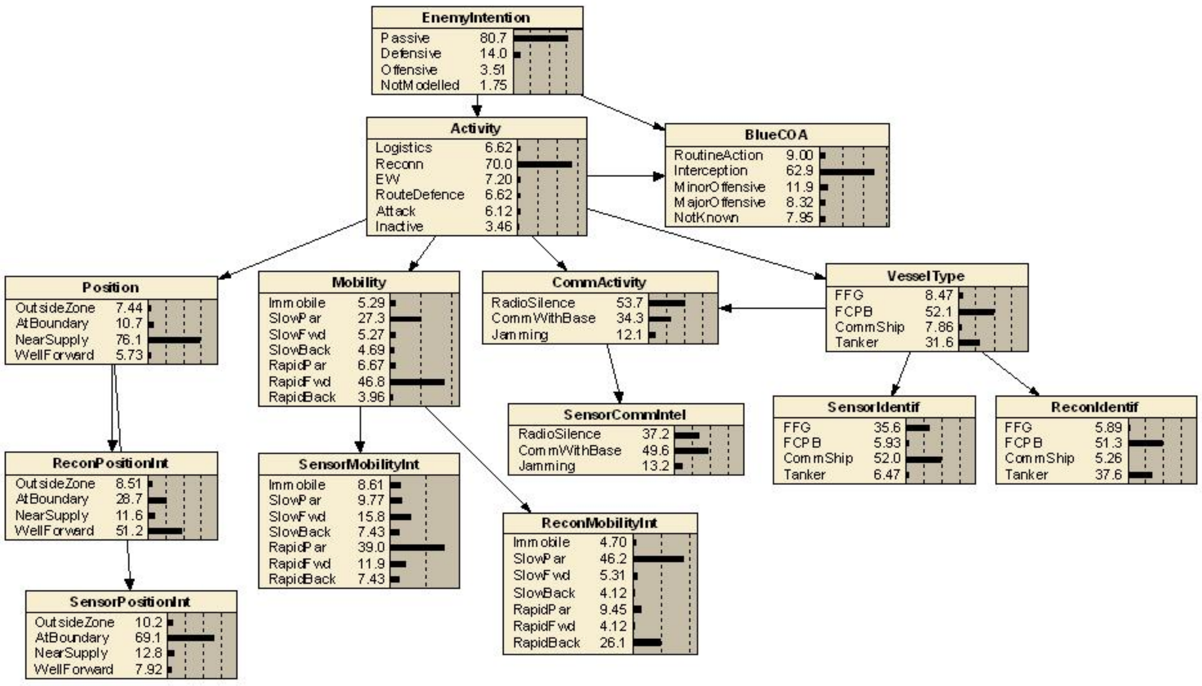
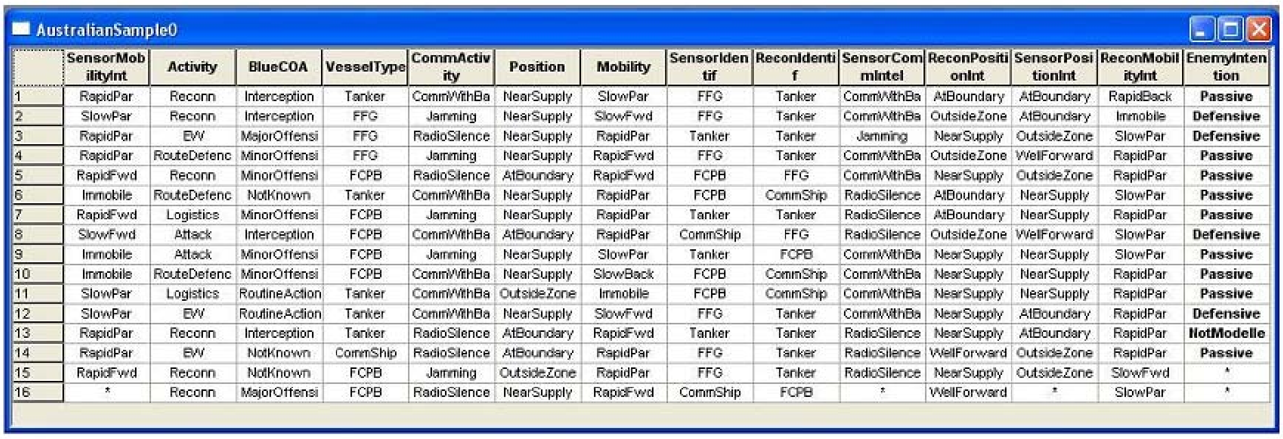


Fig. 14 – Sample BBN for an Australian Naval Exercise


AustralianSample0

	SensorMobilityInt	Activity	BlueCOA	VesselType	CommActivity	Position	Mobility	SensorIdentif	ReconIdentif	SensorCommIntel	ReconPositionInt	SensorPositionInt	ReconMobilityInt	EnemyIntention
1	RapidPar	Reconn	Interception	Tanker	CommWithBa	NearSupply	SlowPar	FFG	Tanker	CommWithBa	AtBoundary	AtBoundary	RapidBack	**Passive**
2	SlowPar	Reconn	Interception	FFG	Jamming	NearSupply	SlowFwd	FFG	Tanker	CommWithBa	OutsideZone	AtBoundary	Immobile	**Defensive**
3	RapidPar	EW	MajorOffensi	FFG	RadioSilence	NearSupply	RapidPar	Tanker	Tanker	Jamming	NearSupply	OutsideZone	SlowPar	**Defensive**
4	RapidPar	RouteDefenc	MinorOffensi	FFG	Jamming	NearSupply	RapidFwd	FFG	Tanker	CommWithBa	OutsideZone	WellForward	RapidPar	**Passive**
5	RapidFwd	Reconn	MinorOffensi	FCPB	RadioSilence	AtBoundary	RapidFwd	FCPB	FFG	CommWithBa	NearSupply	OutsideZone	RapidPar	**Passive**
6	Immobile	RouteDefenc	NotKnown	Tanker	CommWithBa	NearSupply	RapidPar	FCPB	CommShip	RadioSilence	AtBoundary	NearSupply	SlowPar	**Passive**
7	RapidFwd	Logistics	MinorOffensi	FCPB	Jamming	NearSupply	RapidPar	Tanker	Tanker	RadioSilence	AtBoundary	NearSupply	RapidPar	**Passive**
8	SlowFwd	Attack	Interception	FCPB	CommWithBa	AtBoundary	RapidPar	CommShip	FFG	RadioSilence	OutsideZone	WellForward	SlowPar	**Defensive**
9	Immobile	Attack	MinorOffensi	FCPB	Jamming	NearSupply	SlowPar	Tanker	FCPB	CommWithBa	NearSupply	NearSupply	SlowPar	**Passive**
10	Immobile	RouteDefenc	MinorOffensi	FCPB	CommWithBa	NearSupply	SlowBack	FCPB	CommShip	CommWithBa	NearSupply	NearSupply	RapidPar	**Passive**
11	SlowPar	Logistics	RoutineAction	Tanker	CommWithBa	OutsideZone	Immobile	FCPB	CommShip	CommWithBa	NearSupply	NearSupply	RapidPar	**Passive**
12	SlowPar	EW	RoutineAction	Tanker	CommWithBa	NearSupply	SlowFwd	FFG	Tanker	CommWithBa	NearSupply	AtBoundary	RapidPar	**Defensive**
13	RapidPar	Reconn	Interception	Tanker	RadioSilence	AtBoundary	RapidFwd	Tanker	Tanker	RadioSilence	NearSupply	AtBoundary	RapidPar	**NotModelle**
14	RapidPar	EW	NotKnown	CommShip	RadioSilence	AtBoundary	RapidPar	FFG	Tanker	RadioSilence	WellForward	OutsideZone	RapidPar	**Passive**
15	RapidFwd	Reconn	NotKnown	FCPB	Jamming	OutsideZone	RapidPar	FFG	Tanker	RadioSilence	NearSupply	OutsideZone	SlowFwd	*
16	*	Reconn	MajorOffensi	FCPB	RadioSilence	NearSupply	RapidFwd	CommShip	FCPB	*	WellForward	*	SlowPar	*

Fig. 15 – Sample information system in Rosetta for an Australian Naval Exercise

All fourteen nodes from the Bayesian network are mapped onto attributes of an information system. In each time instant, depending on the frequency of measurements in the decision making process, a new case (an object with fourteen attributes) is created. The values of respective attributes may be obtained directly by the measurement process, or from a BBN if necessary. For example, the first attribute in Figure 15, *SensorMobilityInt*, corresponds to the node of the same name in the BBN in Figure 14, and has a value of *RapidParallel*. If some measurements are missing, this is illustrated by an asterisk in Figure 15.

The operation of software tools to conduct this process in real time is illustrated in Figure 16, with evidence meaning the new sensor measurements or reconnaissance data. Such process can be easily automated with existing tools, since a Netica version exists that has a Java API and can read cases from a text file. In turn, Rosetta, which also has a command language interface, can export its tables as text files to be grabbed by Netica. With a converter software reading Rosetta files, making respective adjustments if some data are missing, and transforming them to the Netica format, the whole system shown in Figure 16 can operate smoothly and enhance the decision making process in real time.

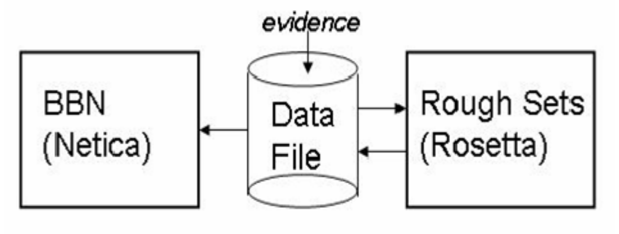


Fig. 16 – Real-Time operation of a BBN tool with a rough set tool.

5. SUMMARY AND CONCLUSION

This paper discussed basic concepts of Bayesian belief networks and rough sets, and showed how they can be combined to enhance the process of reasoning under uncertainty in case of missing values of certain attributes of objects. Bayesian networks and rough sets are individually very adequate tools to solve computational problems with insufficient information and reason about uncertainty. The use of rough sets helps making BBN's more valuable in case of the occasional lack of evidence. It becomes particularly important, when BBN's are used in applications such as real-time decision making or active safety diagnostics, with information being supplied to the nodes during operation. In such cases, losing the source of information for one of the BBN nodes impairs the inference process in the next steps. Using rough set reasoning helps in keeping the BBN in good standing, disregarding the lost source of information.

This logic of this process is very similar to the use of a Kalman filter [50], when the information about the system is updated based on its previous behavior. However, in case of rough sets the information does not have a statistical nature, as in the case of Kalman filtering. Comparing the concepts outlined in this article with the operation of a Kalman filter would be a good topic for further study.

There are several important questions still to be addressed. For example, to apply this method in practice, one would need to know how computationally intensive are the rough set calculations? It seems that for typical applications of Bayesian belief networks, which are used in decision support systems, the deadlines for completing the computations are most likely in the order of minutes or hours, so this issue should not cause problems.

6. ACKNOWLEDGEMENT

The authors gratefully acknowledge contribution to this paper of the late Dr. Henry Pfister.

Part of this work has been funded by a grant SBAHQ-10-I-0250 from the U.S. Small Business Administration (SBA). SBA's funding should not be construed as an endorsement of any products, opinions, or services. All SBA-funded projects are extended to the public on a nondiscriminatory basis.

7. REFERENCES

[1] N.E. Fenton, M. Neil, The use of Bayes and causal modelling in decision making, uncertainty and risk, *Agena Risk White Paper*, June 2011. Available at: http://www.agenarisk.com/resources/white_papers/fenton_neil_white_paper2011.pdf.

[2] F.V. Jensen, T.D. Nielsen, *Bayesian Networks and Decision Graphs. Second Edition*, Springer-Verlag, 2007.

[3] W.J. Dawsey, *Bayesian belief networks to integrate monitoring evidence of water distribution system contamination,* Master Thesis, University of Illinois at Urbana-Champaign, February 2012.

[4] M. Azhdari, N. Mehranbod, Application of Bayesian belief networks to fault detection and diagnosis of industrial processes, *Proc. ICCCE 2010, Intern. Conf. on Chemistry and Chemical Engineering*, Kyoto, Japan, August 1-3, 2010, pp. 92-96.

[5] P. Newham et al., Fog forecasting at Melbourne airport using Bayesian networks, *Proc. Fourth Intern. Conf. on Fog, Fog Collection and Dew*, Santiago, Chile, July 22-27, 2007, pp. 291-294.

[6] N. Fenton et al., Predicting software defects in varying development lifecycles using Bayesian nets, *Information and Software Technology*, (49) 1 (2007), pp. 32-43.

[7] C. Lee et al., Inferring certification metrics of package software using Bayesian belief networks, *Proc. ICIC 2006 – Intern. Conf. on Intelligent Computing*, Kunming, China, August 16-19, 2006, pp. 915-920.

[8] A. Leger et al., Predicting hospital admission for Emergency Department patients using a Bayesian network, *Proc. AMIA Annual Symposium*. Washington, DC, October 22-26, 2005, p. 1022.

[9] S. Ferrari, A. Vaghi, Multisensor fusion for landmine detection by a Bayesian network approach, *Proc. ECSC – 3rd European Conf. on Structural Control*, Vienna, Austria, July 12-15, 2004.

[10] A.K.T. Hui et al., A Bayesian belief network model and tool to evaluate risk and impact in software development projects, *Proc. RAMS 2004 – Annual IEEE Reliability and Maintainability Symposium*, Los Angeles, Calif., January 26-29, 2004, pp. 297-301.

[11] M. Neil, B. Mancom, R. Shaw, Modelling an air traffic control environment using Bayesian belief networks, *Proc. ISSC'03 – 21st Intern. System Safety Conf.*, Ottawa, Ontario, August 4-8, 2003.

[12] K. Sachs et al., Bayesian network approach to cell signaling pathway modeling, *Science STKE*, 148, PE38, 2002.

[13] A. Helminen, *Reliability Estimation of Safety-Critical Software-Based Systems Using Bayesian Networks*, Report STUK-YTO-TR 178, Radiation and Nuclear Safety Authority, Helsinki, Finland, 2001.

[14] G. Dahll, B.A. Gran, The use of Bayesian belief nets in safety assessment of software based systems, *International Journal of General Systems*, (29) 2 (2000), pp. 205-229.

[15] N.E. Fenton, M. Neil, Bayesian belief nets: a causal model for predicting defect rates and resource requirements, *Software Testing and Quality Engineering*, (2) 1 (2000), pp. 48-53.

[16] P. Smets, Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem, *International Journal on Approximate Reasoning*, (9) 1 (1993), pp. 1-35.

[17] J.A. Bernardo, A.F.M. Smith, *Bayesian Theory*, John Wiley and Sons, 1994.

[18] Z. Pawlak, Rough Sets, *International Journal of Information and Computer Sciences*, (11) 5 (1982), pp. 341-356, Available at: http://chc60.fgcu.edu/Images/articles/PawlakOriginal.pdf

[19] Z. Pawlak, *Rough Sets – Theoretical Aspects of Reasoning about Data,* Kluwer Academic Publishers, 1991.

[20] Ji Zhang et al., Neighborhood rough sets for dynamic data mining, *Intern. Journal of Intelligent Systems*, (27) 4 (2012), pp. 317-342.

[21] Y. Li et al., A generalized model of covering rough sets and its application in medical diagnosis, *Proc. ICMLS 2010, Intern. Conf. on Machine Learning and Cybernetics*, Qingdao, China, July 11-14, 2010, pp. 145-150.

[22] M. Bit, T. Beaubouef, Rough set uncertainty for robotic systems, *Journal of Computing Sciences in Colleges*, (23) 6 (2008), pp. 126-132.

[23] G. Ilczuk, A. Wakulicz-Deja, Visualization of rough set decision rules for medical diagnosis systems, *Proc. RSFDGrC 2007 – 11th Intern. Conf. on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, Toronto, Canada, May 14-16, 2007, pp. 371-378.

[24] J. Stefanowski, An empirical study of using rule induction and rough sets to software cost estimation, *Fundamenta Informaticae*, (71) 1 (2006), pp. 63-82.

[25] P.A. Laplante, C.J. Neill, Modeling uncertainty in software engineering using rough sets, *Innovations in Systems and Software Engineering – A NASA Journal*, (1) 1 (2005), pp. 71-78.

[26] Z. Li, G. Ruhe, Uncertainty handling in tabular-based requirements using rough sets, *Proc. RSFDGrC 2005 – Intern. Conf. on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Regina, Canada, 31 August – 3 September, 2005.

[27] R. Wasniowski, A framework for software safety analysis with rough sets, *Proceedings of the 4th WSEAS Intern. Conf. on Software Engineering, Parallel & Distributed Systems*, Salzburg, Austria, February 13-15, 2004.

[28] J.F. Peters, H. Feng, S. Ramanna, Adaptive granular control of an HVDC system: A rough set approach, *Proc. of RSFDGrC 2003 – 7th Intern. Conf. on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Chingqing, China, May 26-29, 2000, pp. 213-220.

[29] N. Meskens, P. Levecq, F. Lebon, Multivariate analysis and rough sets: two approaches for software-quality analysis, *International Transactions in Operational Research*, (9) 3 (2002), pp. 353-369.

[30] L. Shen et al., Fault diagnosis using rough sets theory, *Computers in Industry*, (43) 1 (2000), pp. 61-72.

[31] J. Zalewski, Z. Wojcik, Use of Artificial Intelligence Techniques for Prognostics: New Application of Rough Sets, *Intern. Journal of Computing*, (11) 1 (2012), pp. 73-81.

[32] T. Bayes, Essay towards solving a problem in the doctrine of chances, *Philosophical Transactions of the Royal Society of London*, 53, (1763), pp. 370-418.

[33] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan-Kaufmann, 1988.

[34] Norsys Software Corporation. *Netica. Version 4.08*, Vancouver, Canada, URL: http://www.norsys.com

[35] Hugin Expert A/S. *Hugin Developer Software Package*. Aalborg, Denmark. URL: http://www.hugin.com/

[36] M. Kryszkiewicz, Comparative study of alternative types of knowledge reduction. *International J. of Intelligent Systems*, (16) 1 (2001), pp. 105-120.

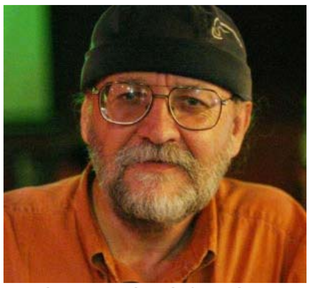
[37] W. Ziarko, Rough sets as a methodology for data mining, In: *Rough Sets in Knowledge Discovery 1: Methodology and Applications*. Physica-Verlag, 1998, pp. 554-576.

[38] J. Komorowski, L. Polkowski, A. Skowron, Rough sets: a tutorial. In: S.K. Pal and A.

Skowron (Eds.), *Rough-Fuzzy Hybridization: A New Method for Decision Making*, Springer-Verlag, 1998.
[39] A. Øhrn, J. Komorowski, J. Rosetta, A rough set toolkit for analysis of data, *Proc. RSSC'97 – Third Intern. Joint Conf. on Information Sciences, Fifth Inter. Workshop on Rough Sets and Soft Computing*, Durham, NC, (3) (1997), pp. 403-407.
[40] J. Grzymała-Busse. LERS – A system for learning from examples based on rough sets. In: R. Słowiński (Ed.), *Intelligent Decision Support: Handbook of Applications and Advances of Rough Set Theory* (pp. 3-18), Kluwer Academic Publishers, 1992.
[41] J. Grzymala-Busse, Three approaches to missing attribute values – a rough set perspective, *Proc. Workshop on Foundation of Data Mining – 4th IEEE Intern. Conf. on Data Mining*, Brighton, UK, November 1-4, 2004.
[42] I.E. Chen-Jimenez, A. Kornecki, J. Zalewski, Software safety analysis using rough sets, *Proc. IEEE Southeastcon'98*, Orlando, Fla., April 24–26, 1998, pp. 15-19.
[43] A. Kornecki, J. Zalewski, Experimental evaluation of software development tools for safety-critical real-time systems, *Innovations in Systems and Software Engineering – A NASA Journal*, (1) 2 (2005), pp. 176-188.
[44] A. Kornecki, J. Zalewski, Software development tools qualification from the DO-178B certification perspective, *Crosstalk – The Journal of Defense Software Engineering*, (19) 4 (2006), pp. 19-22.
[45] M. Neil, N. Fenton, Predicting software quality using Bayesian belief networks, *Proc. SEW-21 – Annual NASA Goddard Software Engineering Workshop*, Washington, DC, December 4-5, 1996, pp. 217-230.
[46] B.A. Gran et al., Estimating dependability of programmable systems using BBNs, *Proc. SAFECOMP 2000 – 19th Intern. Conference on Computer Safety, Reliability and Security*, Rotterdam, The Netherlands, October 24-27, 2000, pp. 309-320.
[47] K.A. Delic, F. Mazzanti, L. Strigini, Formalising engineering judgement on software dependability via belief networks, *Proc. DCCA-6 – 6th IFIP Intern. Working Conf. on Dependable Computing for Critical Applications*, Garmisch-Partenkirchen, Germany, March 5-7, 1997, pp. 291-305.
[48] A. Kornecki, N. Brixius, J. Zalewski, *Assessment of Software Development Tools for Safety-Critical, Real-Time Systems*, Report DOT/FAA/AR-06/36, Federal Aviation Administration, Washington, DC, 2007.
[49] B. Das, *Representing Uncertainties Using Bayesian Networks*, Report DSTO-TR-0918, Defence Science and Technology Organisation, Electronics and Surveillance Research Laboratory, Sydney, Australia, 1999.
[50] R.G. Brown, P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering, Third Edition*, John Wiley and Sons, 1997.

__Andrew J. Kornecki__ is a professor at the Department of Electrical, Computer, Software and System Engineering at the Embry Riddle Aeronautical University. He has more than 20 years of research and teaching experience in areas of real-time computer systems. He has been conducting industrial training on real-time, safety-critical software in medical and aviation industries and for the FAA Certification Services. Recently, he has been engaged in work on certification issues and assessment of development tools for real-time, safety-critical systems.

__Slawomir T. Wierzchon__ is a professor of computer science at the Institute of Informatics of the University of Gdansk in Poland, and at the Institute of Computer Science of the Polish Academy of Sciences. He has been the Program Chair of multiple international conferences and serves on the Editorial Board of the International Journal of Biometrics. His research interests include computational intelligence, biologically inspired computations, machine learning, data analysis, and spectral graph theory.



__Janusz Zalewski__ is a professor of Computer Science and Software Engineering at Florida Gulf Coast University, in Ft. Myers, Florida, USA. He previously worked at nuclear research labs in Europe and the U.S. and consulted for the government and industry. He also had fellowships at NASA and Air Force Research Labs. His research interests include real-time embedded and cyberpysical systems, prognostics of complex systems, and software engineering education.