International Journal of Reliability, Quality and Safety Engineering
Vol. 13, No. 1 (2006) 61–72
© World Scientific Publishing Company

ON USING SOFT COMPUTING TECHNIQUES IN SOFTWARE RELIABILITY ENGINEERING

HENRIK MADSEN[*] and POUL THYREGOD[†]
IMM, Technical University of Denmark
Building 321 — Richard Petersens Plads
Kgs. Lyngby, 2800, Denmark
[*]*hm@imm.dtu.dk*
[†]*pt@imm.dtu.dk*

BERNARD BURTSCHY
Telecom Paris, ENST, 46 rue Barrault
75634 Paris Cedex 13, France
burtschy@enst.fr

GRIGORE ALBEANU[‡] and FLORIN POPENTIU[§]
UNESCO Chair in Information Technologies
University of Oradea, 1 University Str
Oradea, 410087, Romania
[‡]*galbeanu@netscape.net*
[§]*popentiu@imm.dtu.dk*

Received 15 September 2005
Revised 15 December 2005

Previous investigations have shown the importance of evaluating computer performances and predicting the system reliability. This paper considers soft computing techniques in order to be used for software fault diagnosis, reliability optimization and for time series prediction during the software reliability analysis. It is shown that the study of the data collections during a software project development can be done within a soft computing framework.

Keywords: Software reliability; soft computing; reliability optimization.

1. Introduction

Soft computing is tolerant to imprecision, uncertainty, partial truth, and approximation. Fuzzy Logic and Probabilistic Reasoning — based on knowledge-driven reasoning, also called *approximate reasoning*, Neural Computing and Evolutionary Computation — as *data-driven search* and *optimization* approaches — are the principal constituents of the soft computing field. These constituents, which are complementary rather than competitive, are used in this paper to describe and manage different aspects in software reliability engineering. Soft computing is a term originally coined by Zadeh[1] to denote systems that "... exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality."

In general, software reliability is mainly stated in a probabilistic framework and statistical methods are used in an inferential process.[2–4] If there is not a reasonable model or if the proposed model is not adequate then the results of the statistical inference process could become useless or even misleading. It is the aim of this paper to describe more soft computing models and algorithms for software fault diagnosis, reliability optimization and for time series prediction during the software reliability analysis. Also the study of the data collections during a software project development is better to be done in a soft computing manner.[5–7] A software framework that supports uncertainty in the data mining process is proposed. For software reliability optimization both fuzzy models and evolutionary algorithms are used.

2. Background

Vagueness can frequently be associated to many day-to-day processes mainly when human evaluation, observation and decision are evolved. The approximate reasoning is used to manage such situations. Two approaches can be used: probabilistic reasoning and fuzzy logic.

Probabilistic computing can be divided into two classes: *single*-valued and *interval*-valued systems. Bayesian Belief Networks are a typical example of single-valued probabilistic reasoning systems. In general, probabilistic reasoning systems have exponential complexity, when we need to compute the joint probability distributions for all the variables used in a model. Dempster-Shafer (DS) systems are a typical example of interval-valued probabilistic reasoning systems. They provide lower and upper probability bounds instead of a single value as in most Bayesian Belief Networks cases. The DS theory was developed independently by Dempster[8] and Shafer.[9] It is important to note that randomness and fuzziness deal with two different types of uncertainty. In randomness, the uncertainty appeared from the non-deterministic membership of a point from a sample space (of all possible values for a random variable), into a well-defined region of that space — the event states. A probability value describes the tendency or frequency with which the random variable takes values inside the region. In fuzziness, the uncertainty is derived from the deterministic but partial membership of a point — considered in a crisp set, into an imprecisely defined region of that space. Fuzzy logic consists of the theory of fuzzy sets and possibility theory, and it was introduced by Zadeh[10] in order to represent and manipulate data that were not precise but rather fuzzy. Fuzzy sets are defined as functions that map a value, which might be a member of a set, to a number between zero and one, indicating its actual degree of membership. A degree of zero means that the value does not belong to the set and a degree of one means that the value is completely representative of the set. A partial membership value does not represent a frequency. More likely, it describes the degree to which that particular element of the universe of discourse satisfies the property that characterizes the fuzzy set.

Neural networks are introduced in 1943, when McCulloch and Pitts[11] showed that a network of binary decision units could implement any logical function. Then Rosenblatt proposed a one-layer feedforward network, called a perceptron, and demonstrated that it could be trained to classify patterns in linear partitions. By introducing multi-layer neural networks the non-linear separation was also possible. An important problem with the neural networks is the necessity for training. It was proved[12] that a three-layer neural networks (with one input layer, one hidden layer of squashing units, and one output layer of linear units) is an universal functional approximator. From the topological point of view, the neural networks can be classified into feed-forward and recurrent networks. The supervised feed-forward networks include single- and multiple-layer perceptrons, as well as radial basis functions networks. The recurrent networks cover competitive networks, self-organizing maps, Hopfield nets, and adaptive resonance theory models. Such networks are used in unsupervised mode. It must be mentioned that the functional equivalence is proved between radial basis function networks and fuzzy inference systems.[13] Consequently, feed-forward neural networks are mostly used in applications. They are composed of a network of processing units, called also *neurons*. Each neuron performs a weighted sum of its input, using the resulting sum as the argument of a non-linear activation function. By using a training set that samples the relation between inputs and outputs, and a learning method that trains their weight vector to minimize a quadratic error function, neural networks offer the capabilities of a supervised learning algorithm that performs fine-granule local optimization.

Evolutionary computing algorithms provide an adaptive behavior that allows them to handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure. These algorithms are very robust to time-varying behavior, even though they may exhibit low speed of convergence. Evolutionary computing covers many important families of stochastic algorithms, including evolutionary strategies, evolutionary programming and, genetic algorithms. Genetic algorithms searching mechanism starts with a set of solutions called *population*. One solution, in this set, is called a *chromosome*. The search is guided by a survival of the *fit-test principle*. The search proceeds for a *number of generations*. For each generation, based on the fitness function, the fitter solution will be selected to form a new population. During the process, three main operators can be applied: *reproduction*, *crossover* and *mutation*. Reproduction process consists in the combination of the evaluation and selection activities. It copies an individual from one generation to the next. Crossover takes two or more chromosomes and by swapping information between them will produce one or more chromosomes (the so called *children*). Mutation is the process that randomly modifies a part of chromosome's information. The whole cycle can be repeated along a number of generations until certain termination criteria are met (the number of generations is maximally admitted, a high fitness value is obtained or, the simulation time hits

its upper boundary). This chain have to be adapted depending on the particular problem to be solved. An example of using genetic algorithms is given by Matrz.[14] The approach can be extended to support the model developed by Zequeira.[15]

In order to capture the uncertainty and imprecision during activities concerning information systems' design, fuzzy logic theory helps to restore the integrity of the whole system reliability. The approaches are illustrated on two software projects: a decision support system for dynamic control pollution in cements plants,[16] and an optimization software system, useful in quality improvement mainly.

3. Soft Computing Techniques in Software Reliability Modeling

3.1. *Fuzzy modeling*

Fuzzy logic (FL) has proven to be capable of modeling highly nonlinear and multidimensional processes. In general, nonlinear model identification is used in a probabilistic framework for data analysis during the software life cycle. However the uncertainty and imprecision can be captured in a fuzzy manner.

In order to identify a fuzzy model, three generic steps have to be followed: structure identification, parameter estimation and model validation. For the structure identification of the fuzzy model the following elements are necessary: the model type; the number of rules, the positioning of the fuzzy sets in the domain of each variable. Model structure is described by the number of the arrangements of membership functions associated to the fuzzy model. The simplest possible model structure uses one membership function per input variable. The multiple correlation coefficient is computed for the testing data set, and then the model complexity is incrementally increased by adding membership functions. The model structure having the highest correlation coefficient will be selected as an optimal one.

An association between model parameters and the conclusions of the rules is established. The parameter estimation is based on the least square approach, and from this point of view, fuzzy model identification is close to classical nonlinear model identification. The model validation requests checking against domain experts. The most difficult part consists of structure identification. Different approaches were already proposed: when appriori knowledge is available the fuzzy partition can be established by non-linear optimization, otherwise a preliminary data clustering is necessary.[17] Such considerations are useful when a software architecture for reliability monitoring will be implemented in a soft computing framework. Fuzzy modeling, as described above, can be used in order to extend the EV approach introduced by Popentiu-Vladicescu *et al.*[6]

Another usage of fuzzy logic deals with software fault diagnosis consisting of the following tasks: fault detection and isolation, and fault analysis. For an efficient fault detection and isolation the software engineer has to perform the following steps: the generation of messages which reflect the faults, and the decision making concerning the type and time of occurrence of the fault. Three categories of methods for fault detection and isolation are used. These are based on: messages, analytical models, and knowledge. The messaging approach is not a portable one. A change in operation environment may cause false alarm, or masking of faults. Developing analytical models is not an easy task. Only an already known behavior, that is a deterministic process, can be monitored. The knowledge based approach is a fuzzy logic one. This strategy offers the opportunity to combine heuristic knowledge with any model knowledge which may be available.

The decision making process can be based on fuzzy logic according to the following ideas: (a) establishing fuzzy adaptive thresholds that vary according to the operating environment; (b) including a fuzzy classification to provide degree-of-membership to each of the predetermined faults; (c) decide by fuzzy inference consisting of processing rules according to some facts. However, implementing a fuzzy-inference system for fault detection and isolation is a great provocation. The rule base will be determined from inspection of the data according to all users' preferences, or software requirements as defined in the first phase of the software life cycle.

The detection of faults can also be realized by testing and debugging. According to Zeephongsekul,[18] "present SRGMs that incorporates the debugging process assume that when a failure occurs, a debugging effort takes place which may or may not remove the fault with some unknown but estimable probabilities." However, there are a large variety of bugs in a software and the debugging activity should be treated as a fuzzy process rather than crisp. This is why a fault is debugged according to a set of membership functions labelling the debugging performance as: total imperfect debugging, more or less imperfect debugging, less than perfect debugging, . . . , total perfect debugging. In this context some software reliability measures can be proposed: the total fuzzy number of faults; the total fuzzy number of faults that remains in the software, and the expected proportion of faults removed from the software. A measure of variability which incorporates both the imprecision of fuzzy debugging and the randomness due to the number of failures encountered is proposed by Zeephongsekul.[19] These approaches are considered to be implemented and tested to estimate the reliability of the software PoLogCem.[16]

3.2. *Neural networks*

As described above, a neural network is designed to model or simulate the way in which the brain performs a specified task. In general, neural networks are used in optimization processes or time series forecasting. In software reliability, the neural networks can be used in many ways. Let us mention that the neural networks are used to capture the behavior of the general reliability models.[6] Also, a dynamical approach can be used to design a neural network for software reliability prediction based on the approach proposed by Karuanithi *et al.*[20]

Considering the approach given by Popentiu-Vladicescu *et al.*,[5] and the ideas of Cai *et al.*,[21] the investigation concludes positive. It is possible to combine fuzzy logic and neural networks in a neuro-fuzzy system for software reliability growth modeling investigation. In this context we describe the methodology for controlling neural networks by fuzzy logic. Firstly, the inputs to the neuron and/or the weights must be fuzzy sets. A fuzzyfication procedure has to be applied to satisfy this requirement. There are three possibilities: (1) real inputs and fuzzy weights; (2) fuzzy inputs and real weights; (3) fuzzy inputs and fuzzy weights. Such a combination introduce the idea of hybrid computing[22] for software reliability modeling. Let us mention that neural networks, as proposed by Chambers and Mount-Campbell,[23] can be used in a hybrid computing based optimization process.

3.3. *Evolutionary computing*

Evolutionary computing is used for solving optimization problems. As mentioned by Kuo and Prasad,[24] the evolutionary computing "can be viewed as an adaptation of a probabilistic approach based on principles of natural evolution." Practically, this approach covers many important families of stochastic algorithms, including evolutionary strategies, evolutionary programming and, genetic algorithms (GAs). Each of them manages a population of trial solutions, imposes random changes to those solutions, and incorporates selection to determine which solutions will be included in future generations. Evolutionary strategies and evolutionary programming approaches emphasize mutational transformations that maintain behavioral linkage between each parent and its offspring.

For the system-reliability optimization, a lot of research concerning the usage of genetics algorithms are already done. As already mentioned, the genetic algorithms emphasize models of genetic operators as observed in nature, such as crossing-over, inversion, and point mutation, and apply these to abstracted chromosomes, like binary sequences or chains of attributes. For complex distributed software systems, evolutionary computing approach can be used for redundancy allocation in order to optimize the software reliability under some cost constraints. However, this is a meta-heuristic search. They are easily designed and implemented, but the determination of appropriate values for the parameters which control the evolutionary process is difficult. If the penalty for infeasibility another controlling parameters are not selected properly, these algorithms converge to a local optimum or will converge to the global optimum, but very slow.

However, the problem can be solved using fuzzy logic which will provide dynamic control of a genetic algorithm resources (population size, selection impact, probabilities of crossover and mutation) for improving the convergence performances. As an example, the Rule-base will contain fuzzy-rules such: If (average fitness/best fitness) is LARGE then population size should INCREASE, and If (worse fitness/average fitness) is SMALL then population size should DECREASE. Another usage of the evolutionary computing, in special the case of genetic algorithms, is the training of neural networks. Genetic algorithms can be used in developing the neural network architecture (establishing the number of hidden layers, nodes within the layers, the connectivity), or to optimize the weights for a given architecture. According to Shapiro,[25] Genetic algorithms tune fuzzy systems either by adapting the fuzzy membership functions and/or facilitating the learning of the fuzzy if-then rules. The above considerations show the effort to combine neural networks, fuzzy logic and evolutionary computing technologies. We appreciate that the merged technologies will be effective if processing information for classical and intelligent databases is considered and special methods for data mining are implemented.

4. A Soft Computing Framework for Software Reliability Engineering

The software reliability forecasting is a problem of increasing importance mainly for safety-critical applications. A software architecture for monitoring the reliability in distributed systems, called SADS, was proposed by Popentiu-Vladicescu and Sens.[26] The main objective of SADS is to improve the software architecture for monitoring fault-tolerant design in distributed systems. The proposed technique is based on a chain of automatic data collection, which allows us the possibility to adjust, during the execution, the strategy of fault management. Considering the above reasons, this architecture has to be updated in order to include fuzzy approaches that will be applied during all stages in the software development. This is the reason to include fuzzy rules and organize the collected information both in a relational database and an intelligent database. A module, called RKD, to discover the knowledge about patterns in time series and data collections explored in an explanatory approach is included. This module has two main functions: a verification oriented data mining-for verifying user's hypothesis, and a discovery-oriented data mining-the system will find new rules and patterns autonomously. This approach is established according to the ideas of Fayyad and Uthurusamy.[27]

The knowledge obtained by the descriptive method implemented by RKD is used for visualizations, and the knowledge obtained by predictive methods implemented by RKD is used in regression and classification (cluster analysis). We appreciate that data are subject to some noise so a statistical measure of accuracy is associated to each rule obtained in the statistical data mining process. For the fuzzy data mining process, a degree of membership is associated with every fuzzy rule.

The previous SADS architecture[6] integrates a MONITOR-module, a STAT-module, a PRED-module and a SEL-module. The present proposal extends the previous architecture by including two new modules: the RKD module and the OPTIM model. The functionalities of the previous modules are extended. The RKD (Reliability Knowledge Discovery) module is connected to all previously considered modules. The MONITOR module allows the collecting of the information about the evolution of the whole system, including the environment such as the use of resources in terms of memory, files and communication. A specific host, namely *the collector*, will collect information from remote involved hosts.[26] The collector will be subscribed to specific information chosen by a "reliability administrator". On each host a server called *the monitor agent* will collect information concerning local processes and will send it back to the collector. Servers rely on a high-level communication layer, the Network Message Server. The RKD module, also, searches information for testing hypothesis about the system and tries to discover the new behavior of the system. The module of SADS called "STAT-statistical approach" allows filtering the data supplied by the former module and produces a model that can be used for prediction. The structure of this module includes both parametric and non-parametric statistical resources and artificial Neural Networks utilizing information on the state of the operational environment such as proportional hazards.

The extended STAT module integrates fuzzy regression and fuzzy approaches in time series processing. The PRED-module makes predictions about the future failure behavior of the application and the operational environment. Moreover, the predictions are used to calibrate the statistical models. This module will be extended in future to support fuzzy forecasting. The SEL-module uses the information of the prediction module to select the most appropriate algorithm for adaptability of fault management. A comparative analysis of the predictions allows us to choose between the pessimistic approach to favor the recovery delay and the optimistic one to reduce the cost of fault management under the failure — free execution. The module, called OPTIM, is dedicated to optimal reliability allocation for large software projects. This module extends the functionalities using the aproaches considered by Popentiu and Albeanu.[28] There are included evolutionary strategies in solving the optimization problems formulated in this reference. Let us mention that two kind of optimization problems are considered: (1) to minimize the total cost of achieving a target reliability, and (2) to maximize the system reliability subject to a budget constraint. The used methods in cost estimation are the COCOMO model and the Putnam's SLIM model.[29] When a separable model is used then the total cost function is a sum of the components' cost. When a nonseparable model is considered a non-convex function is obtained and genetics algorithms are useful to find a global minimum.

The soft computing framework outlined in this section is a computer aided support decision for software reliability engineering, supporting a client-server architecture. It can be used for both workstations and for distributed systems.

5. Practical Experience

The approaches described above are used to investigate the reliability of two software projects: PoLog-Cem 1.01 and FGCemQ 1.0 developed under an acceptance procedure based on the standards family EN-5012X, the standard IEC 61508, the ESA, IAEA, IEEE and NASA recommendations.[30]

PoLogCem 1.01 is a software system for pollution control in cement plants. The structure and the functionalities of this complex software system were previously reported.[16] This project was designed under a quality assurance plan, that means, for every phase an evaluation report — elaborated by the program monitoring staff (an external entity) — provided the recommendations necessary to consider for obtaining a high quality software product.

FGCemQ 1.0 is a software program to be used for the optimization of the cement quality by fuzzy techniques and genetic algorithms. The FGCemQ[31] software applies the idea that the fuzzy logic programming can be used as a way of relating a set of outputs to a set of inputs in a so-called "model-free" way. The FL module comprises the following three main components: (1) a fuzzification interface: the entrance data are fuzzified and transformed into vague information (mainly based on a crisp-fuzzy converter[32]), (2) a fuzzy set of rules: the "if ... then ..." set of rules is established (the so-called the fuzzy rule base) and the vague results are obtained and, (3) a defuzzification interface: the vague results are transformed into firm results (by center of gravity, mean of maxima, center of area or variants). This approach is a fuzzification of the strategy used in the optimization module of the PoLogCem software, designed, implemented and reported by Madsen *et al.*[16] The GAs module of the FGCemQ software represents a way to perform a randomized global search in a solution space.[31]

Preliminary results concerning the usage of the above mentioned soft computing approaches applied to the mentioned software projects, show that: (1) Manually collecting data is a difficult task and an automatically monitoring software can be a very good choice. (2) Software reliability experience can be included as basic knowledge in an intelligent data base with a fuzzy approach; (3) Fuzzy rules must be proposed by a human expert assisted by a discovery-knowledge software module by data mining techniques; (4) A hybrid soft computing approach can be used; (5) A special attention will be paid to time series when a neural approach is tried. It was obtained that the neural network will describe very well what has happened but it fails to forecast what will happen in time. (6) Fuzzy regression can be an alternative in order to forecast the time interval in the fault detection process. (7) An optimization module for reliability allocation is useful. However, it is better to use software engineering techniques to minimize the number of bugs in the software. That means that the design and implementation of the software have to be monitored in order to satisfy the specified quality requirements, and appropriate strategies will be activated, when necessary. (8) For theoretical software reliability point of view, the software reliability growth models used for adaptability of fault management will be selected depending on the results provided by the prediction module. Because a fuzzy prediction can be obtained, in the case of our proposal, a defuzzyfication procedure will be applied to obtain a crisp value to be used for the study of SRGM's behavior.

Finally, we remark a positive reaction of the top management when artificial intelligence techniques are used to investigate the reliability of the software projects under development. We appreciate that such a reaction is a result both of the reliability reports provided to them and of the present impact of the software technologies based on soft computing.

6. Conclusions

This paper investigates the usage of the soft computing techniques for software reliability. Starting with a previous framework, this was extended with new modules and functionalities in order to support fuzzy approaches, evolutionary computing and data mining.

Acknowledgements

This project, partially, has been developed under the *NATO-STI program* within the framework of the collaborating linkage grant *EST.CLG.979542*. The investigations on some soft computing developments for software reliability were supported by an internal research project at *UNESCO Chair in Information Technologies*, University of Oradea.

References

1. L. A. Zadeh, Fuzzy logic and soft computing: Issues, contentions and perspectives, in *Proc. 3rd Int. Conf. Fuzzy Logic, Neural Nets and Soft Computing* (Fuzzy Logic Systems Institute, Iizuka, Japan, 1994), pp. 1–2.
2. G. Albeanu and F. Popentiu, Total quality for software engineering management, in *Handbook of Reliability Engineering*, ed. H. Pham (Springer, London, 2003), pp. 567–584.
3. B. Burtschy, G. Albeanu *et al.*, Improving software reliability forecasting, *Microelectronics and Reliability* **37**(6) (1997) 901–907.
4. J. D. Musa, *Software Reliability Engineering* (McGraw-Hill, New York, 1999).
5. F. Popentiu-Vladicescu, B. Burtschy and G. Albeanu, Time series methods for modeling software quality, in *Proc. European Conf. Safety and Reliability*, eds. E. Zio *et al.* (Politechnico Di Torino, Italy, 2001) **1**, pp. 9–15.
6. F. Popentiu-Vladicescu, G. Albeanu *et al.*, Software architecture for distributed systems (SADS): NN and EV approaches, in *Proc. European Conf. Safety and Reliability*, eds. E. Zio *et al.* (Politechnico Di Torino, Italy, 2001) **2**, pp. 1031–1037.
7. F. Popentiu Vladicescu and G. Albeanu, On a probabilistic-fuzzy model in software reliability engineering, in *Proc. KONBIN03 — The 3rd Safety and Reliability Int. Conf.* (Gdynia, 2003) **3**, pp. 517–523.
8. A. P. Dempster, Upper and lower probabilities induced by a multivalued mapping, *Annals of Mathematical Statistics* **38** (1967) 325–339.
9. G. Shafer, *A Mathematical Theory of Evidence* (Princeton University Press, Princeton, NJ, 1976).
10. L. A. Zadeh, Fuzzy sets, *Information and Control* **8** (1965) 338–353.
11. W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* **5** (1943) 115–133.
12. K. Hornick, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* **2** (1989) 359–366.
13. R. Jang, C.-T. J. Sun and C. Darken, Functional equivalence between radial basis function networks and fuzzy inference systems, *IEEE Trans. on Neural Networks* **4**(1) (1993) 156–159.
14. H. Martz, M. S. Hamada *et al.*, Using genetic algorithms to design Bayesian reliability experiments, in *Proceedings of MMR 2002*, (Trondheim, Norway, 2002). http://www.math.ntnu.no/mmr2002/papers/invited/martz.pdf

15. R. I. Zequeira, A model for Bayesian software reliability analysis, *Qual. Reliab. Engng. Int.* **16** (2000) 187–193.
16. H. Madsen, P. Thyregod *et al.*, A decision support system for pollution control in cement plants, in *Probabilistic Safety Assessment and Management, Proc. PSAM 07 — ESREL'04*, eds. C. Spitzer *et al.* (Springer, Berlin, 2004) **3**, pp. 1784–1789.
17. A. Evsukoff, A. C. S. Branco and S. Galichet, Structure identification and parameter optimization for non-linear fuzzy modeling, *Fuzzy Sets and Systems* **132** (2002) 173–188.
18. P. Zeephongsekul and G. Xia, On fuzzy debugging of software programs, *Fuzzy Sets and Systems* **83** (1996) 239–247.
19. P. Zeephongsekul, On the variability of fuzzy debugging, *Fuzzy Sets and Systems* **123** (2001) 29–38.
20. N. Karuanithi, D. Whitely and K. Malaya, Predictions of software reliability using connectionist models, *IEEE Transactions on Software Engineering* **18**(7) (1992) 563–574.
21. K.-Y. Cai, L. Cai *et al.*, On the neural network approach in software reliability modeling, *J. Systems and Software* **58** (2001) 47–62.
22. P. P. Bonissone, Y. Chen *et al.*, Hybrid soft computing systems: Industrial and commercial applications, in *Proc. IEEE* **87**(9) (1999) 1641–1667.
23. M. Chambers and C. A. Mount-Campbell, Process optimization via neural network metamodeling, *Int. J. Production Economics* **79** (2002) 93–100.
24. W. Kuo and V. R. Prasad, An annotated overview of system-reliability optimization, *IEEE Transactions on Reliability* **49**(2) (2000) 176–186.
25. A. Shapiro, The merging of neural networks, fuzzy logic, and genetic algorithms, *Insurance*: *Mathematics and Economics* **31** (2002) 115–131.
26. F. Popentiu-Vladicescu and P. Sens, A software architecture for monitoring the reliability in distributed systems, in *Proc. ESREL99* (TUM Munich-Garching, 1999), pp. 615–620.
27. U. Fayyad and R. Uthurusamy, Data mining and knowledge discovery in databases, *Communications of the ACM* **39**(11) (1996) 24–26.
28. F. Popentiu-Vladicescu and G. Albeanu, Optimal reliability allocation for large software projects, in *Optim 2002 — The 8th Int. Conf. Optimization of Electrical and Electronic Equipment* (Brasov, Transylvania Uiversity, 2002) **1**, pp. 602–606.
29. N. M. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach* (International Computer Press, PWS Publishing Company, London, 1996).
30. G. Albeanu and F. Popentiu, Software safety concepts. Approval and aceptance of safety critical integrated systems, in *Proc. SIE 2004* (The Romanian Power Enineering Society, Sibiu, 2004), pp. 43–48.
31. H. Madsen, P. Thyregod *et al.*, Considerations concerning a fuzzy — genetic algorithm with application to CEM I type cements quality optimization, in *Proc. 9th World Multi-Conference on Systemics, Cybernetics and Informatics*, eds. Dale Zinn *et al.* (Orlando, 10–14 July 2005) **10**, pp. 350–353.
32. I. Vaduva and G. Albeanu, *Introduction to Fuzzy Modeling* (Bucharest University Press, Bucharest, 2004).

About the Authors

Henrik Madsen received the M.Sc. in Engineering in 1982, and the Ph.D. in Statistics in 1986, both at the Technical University of Denmark. He was appointed Professor in Statistics with a special focus on Stochastic Dynamic Systems in 1999.

He is involved in a number of cooperative projects with other universities, research organizations and industrial partners. His main research interest is related to analysis and modeling of stochastic dynamics systems. This includes signal processing, time series analysis, identification, estimation, grey-box modeling, prediction, optimization and control. The applications are mostly related to Energy Systems, Informatics, Environmental Systems, Bioinformatics, Process Modelling and Finance. He has authored or co-authored approximately 210 papers and technical reports, and about 10 educational textbooks. Also, he is the Head of Center for High Performance Computing at DTU which was opened by the Danish Minister of Research in February 2002.

Poul Thyregod received the M.Sc. in 1966 and the Ph.D. in mathematical statistics in 1970 both from the University of Copenhagen. From 1971 he has been employed as a professor in statistics at Informatics and Mathematical Modeling (IMM) at the Technical University of Denmark. His research activities have been in the field of industrial statistics with focus on quality and reliability issues. He is past president of European Network for Business and Industrial Statistics (ENBIS).

Bernard Burtschy is Professor of Statistics at the National Advanced School of Telecommunications (ENST). Also, he teaches data analysis and prediction methods at the Central Paris School. He is Associate Editor of Communications in Statistics. Also he is the author of numerous articles, and member of France Telecom's Surveys Commission.

Grigore Albeanu received his Ph.D. in Mathematics in 1996 from University of Bucharest. He was appointed Professor of Computer Science at University of Oradea in 2002. From 2004, he is the UNESCO IT Chair holder at University of Oradea. Grigore Albeanu has authored or co-authored more than 60 papers and 10 educational textbooks in applied mathematics and computer science.

Florin Popentiu received the M.Sc. in 1975, and the Ph.D. on reliability in 1984, from the University "Politechnica" of Bucharest. Since 1998 he has been appointed UNESCO Chairholder at the City University, London. At present he is an associate Professor of Software Engineering at UNESCO Department University "Politehnica" of Bucharest. He has been a visiting professor at a number of renowned European universities such as: Telecom Paris, Ecole des Mines Paris, ETH — Zurich and Université Pierre et Marie Curie, Paris. Dr. Popentiu is currently visiting professor at the Technical University of Denmark, and also at the "Grandes Ecoles in Paris". He has published over 100 papers in International Journals and Conference Proceedings and is co-author of 3 books. His research interests focuse on software safety and reliability. He is working on software reliability problems and has been Co-Director of two NATO research projects.