Knowledge-Based Approach to Risk Analysis Modelling

Petr HANÁČEK, Petr PERINGER , Zdena RÁBOVÁ

Department of Computer Science and Engineering,
Faculty of Electrical Engineering and Computer Science
Božetěchova 2, 612 66 Brno
e-mail: { hanacek, peringer, rabova }@dcse.fee.vutbr.cz

Abstract. The article deals with methodology framework for creating risk analysis models of diverse heterogeneous systems. Risk analysis, which refers to the study of threats and their potential impacts, is modelled with the use of knowledge-based models. From a broad range of different knowledge-based approaches we have chosen the fuzzy description for expressing the knowledge base. The fuzzy description suits best the nature of knowledge that we use.

1 Introduction

During design, implementation, and operation of different heterogeneous systems it also might become necessary to choose from different alternatives or options, which can bring some risks. All these risks have to be identified, analysed, and reduced to the minimum by implementation of suitable measures or controls. The process of identification, analysis, and reduction of risks is called *risk analysis*. For risk analysis process automation we proposed the approach of risk analysis modelling.

We are searching an optimal internal structure of the risk analysis model that is capable of expressing a variety of different assets with various values that bring a broad range of vulnerabilities into the system. The *vulnerability* is a weak point in a system that can be exploited and can consequently lead to a malfunction of the system or to a security incident. The intention or potential of the outside world (called environment) to exploit the present vulnerabilities is called a *threat*.

A clear understanding of the concept "risk" is necessary to establish a common frame of reference for further discussions in this paper. For the purpose of this paper, a risk is viewed as:

- the adverse effect if a threat is realised
- the adverse effect if vulnerability is exploited

Viewing a risk with the use of these dual, but parallel, concepts enhances the comprehension of the concept, which is essential for the effective modelling of the risks within a computing facility.

By using the above concept of a risk, risk analysis can be viewed as a systematic examination of:

- all the possible vulnerabilities and the probability that these vulnerabilities will be exploited

- all the possible threats and the likelihood that these threats will be realised

This view of the risk analysis process will now be used to describe a framework for a systematic analysis of computer related risks.

The risk analysis process has three main input values - assets, threats, and vulnerabilities.

Assets

Assets include hardware (computers, memory devices, peripheral equipment), software (operating systems, application programs), data, networks (transmitting equipment and media), and personnel (operators, users, managers). From a risk viewpoint, the assets within a computing environment are interrelated. A vulnerability can affect more than one asset or cause more than one type of loss.

Threats

Threats to all types of assets include people (who may intentionally damage assets), natural events (such as earthquakes, floods, and tornadoes), and accidents (such as fire, burst water pipes, and human mistakes).

Vulnerabilities

Vulnerabilities are physical (unprotected entrances, unreliable environmental control, unreliable power, and weak fire protection). Personnel vulnerabilities are those that can be caused by personnel (mistakes, fraud, theft, blackmail, bribery), and those that can happen to the personnel (injury, death). Hardware vulnerabilities include environmental effects on hardware, failures, and design mistakes. Software has also some vulnerabilities because of the storage media used, and because of unreliability or poor design. Network vulnerabilities include physical and hardware vulnerabilities, as well as insufficient protection against tapping, eavesdropping, and other forms of interception.

The output of the risk analysis process is a set of recommended countermeasures. These countermeasures should be as specific as possible.

As the risk analysis procedure is a creative, intuitive and systematic process, the tools that are used to model this process must likewise be flexible, to be able to reflect the real-world scenario as truly and effectively as possible. Risk analysis often relies heavily on producing numbers, and does not rely sufficiently on human analysis and a common sense to interpret the results. The model must place a minimal hindrance on the intuitive, common sense and creative spirit necessary for the risk analysis process.

As it is not possible to establish the correctness of a risk analysis procedure, it is necessary to ensure that the model is as understandable and comprehensible as possible. Risk analysis should not be a black box system which "magically" provides answers that are to be trusted. It is necessary for the user to be able to intuitively follow the modelling and reasoning the process of the risk analysis, and in this way to verify the correct functioning of the model.

The proposed risk analysis methodology internally contains two models (see Fig. 1). The first one is a knowledge based *behaviour model*. This model is fixed (it does not change in the risk analysis process) and it is based on a *reality knowledge base*. This model describes relations among assets, threats, vulnerabilities, and countermeasures. The second model is a *system model*. This model is system-dependent and it is based on a *system*

knowledge base. It is created in the first phase of the risk analysis process. This model describes the structure of the analysed system and the interdependencies among the assets in the system.

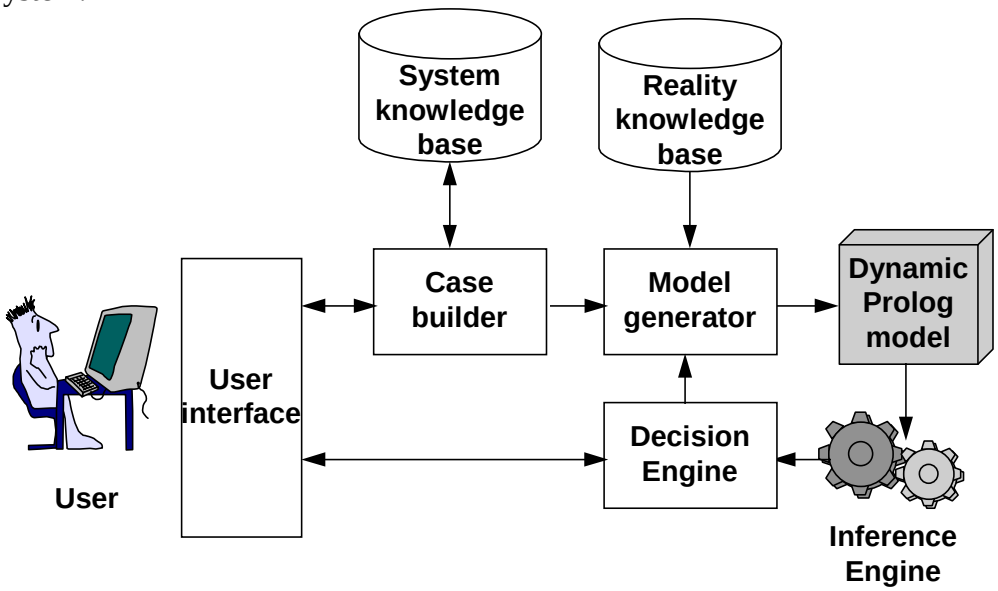

Fig. 1 The structure of the risk analysis system

Most of the above factors are difficult to be quantified and they are usually expressed using vague estimates. The probability of occurrence of threat is not a single value that could be determined with an absolute certainty, but it can be expressed in terms as “low”, “middle”, “high”, or “unknown”. This uncertainty can be modelled in fuzzy logic with the use of fuzzy sets.

2. Design of risk analysis model

Because of the complexity of the problem being solved and due to the incompleteness of the input information, we have decided to build a methodology of risk analysis as an integration of a number of different methods and tools. The basic part of the system is an object-oriented environment of the programming language C++. In this environment, the simulation library SIMLIB/C++ is implemented. The SIMLIB library contains tools for different kinds of system description, including fuzzy logic description.

In a heterogeneous simulation system, we need a fuzzy description each time we do not know an exact description of the model, or we need a simple (fast, but less accurate) solution of the problem.

In SIMLIB/C++, the fuzzy part of the model needs suitable interfaces. The developed fuzzy block has continuous/fuzzy input and output, it can be used as any other continuous block.

The fuzzy representation of a model encapsulated in a fuzzy block should implement the following operations: optional fuzzification of inputs, application of fuzzy operators and implication methods for all the fuzzy rules given, aggregation of outputs from all the rules, and optional defuzzification of outputs. This process is shown in Figure 2 together with the types of the processed data.

- **Fuzzification** is the conversion from a continuous input value (real number) to a fuzzy set value. We use *membership functions* to determine the degree of membership for a given numerical value.
- **If-then rules** are defined by the user in the general form

if (antecedent) consequent

These rules allow computing outputs as the consequence of a user-defined antecedent. The problem here is the partial degree of membership of the antecedent, which should be applied to the consequent with the given weight of the rule. This process is called an *implication*.

Both the antecedent and consequent can have multiple parts. In the antecedent, we can use fuzzy && (AND), || (OR), and ! (NOT) operators. The method of computing the result of an AND operator is usually the minimum of both operands (which are numbers), the OR operator uses a maximum of the operand values, and NOT operator computes *1-x*. There are other algorithms to be used for the fuzzy operator evaluation, too.

- **Aggregation** is the method used for evaluating a single output value (for each output variable) from partial outputs of many if-then rules in the description of a fuzzy block. The output value is computed using one of the suitable algorithms (maximum, sum).
- **Defuzzification** is used for conversion from a fuzzy set to a single numerical value. There are many applicable methods, where most often the method of *centroid* is used. Other possible methods are for example: *bisector, middle of max, largest max, smallest max.*

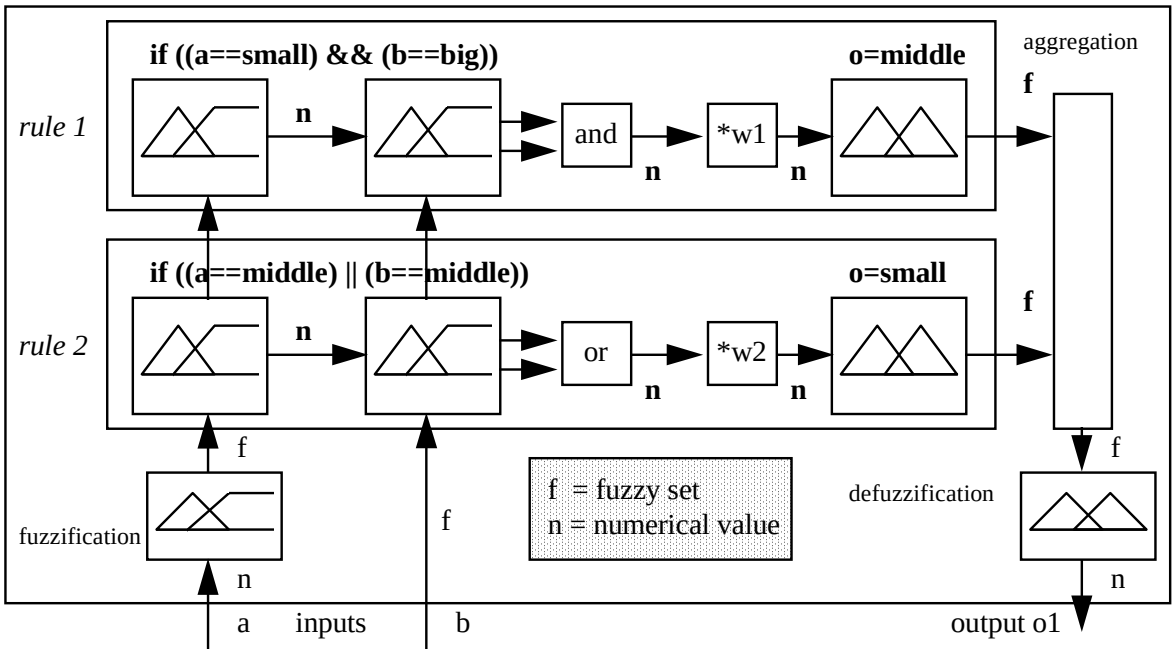

Fig. 2 Fuzzy inference diagram

3. Implementation

We use the C++ language for the fuzzy inference system implementation. This language allows the use of encapsulation, polymorphism, and operator overloading.

Fuzzy set

The class *FuzzySet* implements the description of a fuzzy set value. It contains a reference to a set of membership functions and the associated values in the range <0,1>. Each fuzzy set contains a range of possible numerical values, which can be fuzzified to this fuzzy set.

Fuzzy set membership functions are defined as subclasses of the abstract base class *FuzzyMembershipFunction*. The user can add any kind of membership functions by inheriting this class and defining a suitable methods.

The correct value of a fuzzy set can be created by the fuzzification method or by an assignment. Each membership value can be obtained using operator [].

The class *FuzzyOutput* is defined for a fuzzy output specification. This class has a method for defuzzification and some special behaviour used by the aggregation phase of the fuzzy inference, but other characteristics are inherited from *FuzzySet*.

Fuzzification and Defuzzification

Fuzzification is a method of a fuzzy set. It assigns a new value to a fuzzy set object. If the input value does not fall into the range of possible values, an error is reported.

Defuzzification is implemented as a method of class *FuzzyOutput*. It computes the numerical value of the output. Various defuzzification methods can be used.

Inference rules

The inference rules implementation uses operator overloading. The operator == computes the degree of membership and the operators && (AND), || (OR),! (NOT) allow for rule combinations. All those operators return the object of the class *FuzzyValue*, which represents real number.

We use *FuzzyValue::operator bool* for some actions at the end of the *if-condition* evaluation. Those actions include storing of the condition result in a special variable used later by the operator = and setting the rule weight to 1.0. In the statement after *if(condition)*, we can use the function *weight* to change the weight of the rule. The operator = should be used for an output value specification.

The next examples show the use of overloaded operators:

```
if(in=="small") weight(0.9), out="big";
if(in=="big" || in == "medium") out="small", o2="zero";
if(in=="big" || in == "medium") { out="small"; o2="zero"; }
```

where *in* is a fuzzy input variable, *out* and *o2* are fuzzy output variables, *"small"*, *"big"*, *"medium"* are fuzzy values, which can be members of fuzzy set *in*.

The rules are evaluated in the given order, results are stored to the output variables *out*, *o2* and can be aggregated later. We can use any number of rules. As the output variables should be initialised before the execution of fuzzy rules, we defined the abstract class *FuzzyBlock* with the method *Evaluate*, which is suitable for a fuzzy rules definition. The evaluation of the block (performed automatically) ensures the initialisation, evaluation of inputs, fuzzification, inference of rules, aggregation, and defuzzification.

4. The structure of the knowledge base rules

The expert system used will allow easy orientation in the large amount of information and will compare the results of simulation experiments with similar cases and their consequences that are stored in the knowledge base. The expert system creates new questionnaires with questions for gathering input data for new simulation runs.

The behaviour model knowledge base contains the following types of knowledge:

1. Relevancy of an impact/threat, vulnerability and countermeasure to the asset type, e.g.:

```
"An impact I applies to the asset type A."
```

2. A relation between an impact/threat and the vulnerability, e.g.:

```
"A threat T exploits the vulnerability V with factor x."
```

3. A relation between a countermeasure and vulnerability, e.g.:

```
"A countermeasure C minimises the vulnerability V by factor x."
```

4. A relation between a countermeasure and impact/threat, e.g.:

```
"A countermeasure C minimises the probability of threat T by factor x."
```

These rules have to be expressed in the fuzzy logic notation. The rules of type 1 are exact logic rules that can be expressed in the conventional logic and they do not need the use of fuzzy logic. The rules of type 2, 3, and 4 need the fuzzy logic to be used. Now we will show some examples of fuzzy logic rules:

```
"If Quality_of_password_management is low then
   Probability_of_user_masquerade is high." (type 2 rule)
"If Authentication_calculators_are_used then
   Probability_of_user_masquerade is low." (type 3 rule)
```

4. Conclusion

The design of the described system for the risk analysis of information systems is based on extensive experience of the authors in this area. The experience includes classical modelling, building knowledge based models, and information system security.

Although the areas of simulation of systems and knowledge-based methods were developed separately, we can find a lot of similarities and common principles in both of these scientific areas. These similarities show close methodological relations between these areas and the techniques and tools used.

The SIMLIB library used for an implementation of a fuzzy logic inference engine was previously tested with other applications. This simulation library is not only limited to the fuzzy logic operations, but supports a wide variety of other simulation primitives that can be used for modelling other aspects of target system as well.

References

[1] Courtney, R.: Security risk assessment in electronic data processing, AFIPS Conference Proceedings of the National Computer Conference, AFIPS, Arlington, Va., 97-104

[2] Zadeh, L.A.: Fuzzy Logic, Computer, Vol 1, No. 4, 1988

[3] Hanáček P.: Information System Security Models, XVI. Moravo-Silesian International Colloquium "Selected Problems of Simulation Models", CSS, SCS, EUROSIM, Brno 6.-8.9. 1994, pp. 35-39

[4] Hanáček, P., Rábová, Z.: Processing of Input Data for Risk Analysis, In: Proceedings of Conference ASIS'97, MARQ, Krnov, 1997, s. 91-96, ISBN 80-85988-17-8

[5] Hanáček, P., Rábová, Z.: Knowledge-Based Simulation in Risk Analysis, In: Proceedings of ASIS 1998, MARQ, Krnov, 1998, s. 79-84, ISBN 80-85988-26-7

[6] Jenkins, B.: Security risk analysis and management, Norman Data Defense Systems, Inc., 1995

[7] Ru, W.G., Eloff J.H.P.: Risk analysis modelling with the use of fuzzy logic, Computer & Security, Vol. 15, No. 3, pp. 239-248

This work was supported by the research intention No. CEZ:J22/98: 262200012 - "Research in Information and Control Systems" and by the Grant Agency of Czech Republic grant No. 102/98/0552 " Research and Application of Heterogeneous Models".