Risk–Meaningful Databases

James Bradley
Department of Computer Science
University of Calgary

Department of Computer Science Report Number 2002–713 –16

Abstract *Risk–meaningful databases* are central to one of three important approaches to risk elimination in systems, namely use of both a *risk monitoring procedure* for detection of risk, and a *response procedure* for either eliminating or reducing the risk detected. The risk monitoring procedure uses violations of *risk–meaningful database constraints* to signal the presence of risk. The degree to which risk is can be detected depends on a cumulative measure of the risk–meaningful database constraints available. This cumulative measure may be used in an extended version of the risk equation for systems, in order to quantify the relationship between throughput capacity and risk that has been partly reduced or eliminated by means of a risk monitoring procedure.

Two types of risk–meaningful databases are identified, namely *active rm–databases* and *passive rm–databases*. Passive rm–databases are mostly constant in size, they store data about hazardous entities, and can have any of the structures commonly found in databases. In contrast, active rm–databases grow continually, they store incoming data from monitoring units and sensors, they tend to have a more restricted structure, and the rm–database constraints used for detecting risk depend heavily on *co–relationships*, that is, relationships arising from entity or event relations that have non–primary key attributes in common.

Key words: database, complexity measure, constraint, co–relationship, risk, risk equation, risk monitoring

1. Introduction

Research into risk elimination using *risk monitoring procedures* has shown that a *risk–meaningful database* is an essential component of such a system. A risk monitoring procedure is one of the three methods available for risk elimination and reduction in systems [5]. Risk in this context is risk of loss of system throughput capacity.

In an earlier paper [4], it was shown that the expected throughput capacity of a system in an efficient risky environment is the sum of (a) the basic component KR due to risk−free deployment of the system resources R, where K is a constant (b) the best−case gain component G due to reward for running risk when the hazards risked do not occur, and (c) the loss or risk component L that is the average loss due to hazards actually occurring on occasion.

The mean or expected throughput capacity I is expressed concisely by the risk equation formulations:

$$I = KR + G - L = KR + (c - 1)L$$

or:

$$I = R[K + (c - 1)r(E)] = R[K + br(E)]$$
$$= RK + Rcr(E) - Rr(E)$$

where I is the mean throughput capacity, K is a constant, R is the resources employed, r(E) is the risk per unit R due to the environment E in which the system operates, and c is a constant called the *risk efficiency coefficient*, which is G/L, or the amount of hazard−free gain per unit of average loss due to hazards occurring.

In a set of *efficient environments*, c will be both constant and the largest value possible for operation of the system [4]. Notwithstanding the constant c in such a set of environments, the risk r(E) can vary from one such efficient environment to another. A maximum possible value of c means the most throughput capacity I, for a given level of risk.

The risk r(E) is *MEL−risk* [4, 5], which is *mean expected loss with respect to the best−case throughput capacity*. *SD−risk* may be used when the per period variations in throughput capacity are distributed normally, or near normally. In such a case, SD−risk is close to half of the MEL−risk. SD−risk is standard deviation risk [4, 5], computed *as the standard deviation of per period variations with respect to the mean*.

In [5], we showed how it is possible to improve the throughput capacity even further, by reducing, or eliminating, the mean losses L, or Rr(E), either by adding preventive resources, or by carrying out a precautionary procedure. We also showed that there is another important method of achieving the same result. It is elimination of risk, and thus elimination of mean losses L, by means of application of a system−supported risk monitoring procedure, as part of an overall procedure for coping with the risky environment. The risk monitoring procedure typically involves use of what we have called a *risk−meaningful database*. To understand what this is, we first need to review risk−monitoring procedures

Risk monitoring procedure concepts

Sometimes it is not known in advance where or when the risk of a hazard will be present, that is, we do not know the regions or periods in which the hazard could occur, as opposed to those regions or periods where it cannot occur.

Preventive resources cannot be used, for we do not know where to apply them. Similarly, precautionary procedures cannot be used, for likewise we do not know when to apply them. The classic naval example is a long sea lane, where a torpedo attack could occur anywhere, in one stretch one day, and in another stretch another day, and so on.

In such cases, it is simply known that there will be some periods or regions, variable from one time period to another, where the hazard can occur, that is, where the system is exposed to risk, and that there will be other time periods and regions where the hazard cannot occur and there is no risk.

It is the function of a real time risk monitoring and detection procedure, which is a component of a *risk–environment coping procedure*, to detect the periods or regions where the risk is present, and so generate an alert that triggers a *response procedure*, also part of the risk–environment coping procedure, to take immediate action to eliminate the risk, in whole or in part.

Clearly, we have two components, the *risk detection component*, and the resulting *action or response component*. In some cases, there may be very little doubt that the risk of a hazard occurring is high, if action is not taken. But this near certainly is usually not the case. In practice, the detector will detect only the risk of a hazard, not the certainty of it.

The following example should further clarify this idea, which is crucial for understanding the risk monitoring approach to risk elimination and reduction. Suppose we have a wartime situation, where there is surface ship that might be attacked by a submarine, and the ship is using sonar to detect any submarine in its vicinity. We can assume that, when a submarine is detected, the ship has the means to eliminate the threat. But the ship will likely never be able to detect the submarine for sure, for the sonar may be responding to a whale, or a submerged wreck, or a school of fish, or a clump of seaweed, or driftwood, and so on. Thus the detection system in practice will be able to detect only the risk of a hazard, not the certainty of it.

Another important example is in protecting computers against hostile computer–software agents such as viruses. The virus–detection software, or virus risk–monitoring system, checks incoming files for suspicious bit patterns that could indicate a virus. If such a pattern is detected, the monitoring system's response will be to execute the response procedure, which will likely involve deleting the suspicious file. However, the virus–detection software cannot usually determine for certain that the incoming file is hostile, only that it might well be. Dud signals are always possible.

It is important to grasp that when a monitoring procedure is used, unlike the case of using precautionary procedures, the agent operates the system as if no risk were present, without knowledge of when or where the risk will appear, and most of the time without any significant slowdown of the system. The agent thus operates the system normally, confident that the continuously–operating risk monitoring procedure will detect risk in time to take short–term action to avoid or reduce loss of system throughput capacity.

A simple, everyday example of this is the use of fire alarms in buildings. Occupants normally function as if no fire risk is present, but immediately a fire alarm goes off, for whatever reason, indicating a risk, but not the certainty, of fire, part of the normal response procedure is to evacuate the building.

Two types of response procedure

As we have seen, an environment coping procedure contains two components: (1) a real–time risk monitoring procedure with built–in capability for detecting the presence of risk of throughput capacity loss, due to a possible hazard in the unfolding environment, and (2) a response component equipped with facilities, perhaps brief precautionary procedures, to respond to, and at least partially eliminate, the risk of the hazards detected.

There are two ways in which the response component of the coping procedure can work. The simplest involves some additional resource deployed by the response procedure to eliminate the risk, independently of the operation of the normal system resources R. In such a case, normal system throughput is not affected by the operation of the response procedure, that is, there is no slowdown effect, even if the risk signal is a dud.

The more complex response involves a precautionary procedure that will tie up the resources R for a short period, reducing normal system throughput. The simpler kind of response procedure thus does not slow down the system; the more complex kind does.

An example should clarify this. Returning to the computer virus detection system earlier, assume that the monitoring or detection component of the coping procedure has detected the risk of a virus in a received file. If a separate processor, dedicated to virus elimination, deals with the suspect file, the other processors are unaffected and there is no slowdown effect. On the other hand, if there is only one processor, that processor will be prevented from operating normally while the suspect file is being dealt with [8] There is thus a slowdown effect.

Elimination of risk of loss of throughput capacity by means of a coping procedure, involving both a risk monitoring procedure and response procedures, is a more complex case than either of risk elimination by preventive resources or precautionary procedures alone.

2. Risk–meaningful Databases and Risk Monitoring with No Slowdown Effect

In this paper we are concerned with risk–meaningful databases, and so we assume the simplest response–procedure case, where there is no slowdown effect, since risk–meaningful databases are not any different with the more complex case, where there is a slowdown effect. We can thus employ the simplest version of the risk monitoring equation, when the simplest type of response procedure is used, with no slowdown effect due to temporary diversion of system resources R [5]. The slowdown effect, and more sophisticated versions of the risk monitoring equation, are covered in detail in [5].

The no–slowdown risk–monitoring risk equation states that for a system with an environment coping procedure and resources R, expected throughput capacity I can be increased by increasing a complexity–measure parameter u in a real–time monitoring procedure component of the coping procedure, in accordance with the following modified risk equation formulations:

$$I = RK + G - (1 - [1 - d]M(u))L$$

or: $$I = R[K + [c - (1 - [1 - d]M(u))]r(E)]$$

or: $$I = R[K + [c - (1 - M(u))]r(E)] \qquad \text{when } d = 0.$$

Here u is an independent variable that measures the length of, and thus the complexity of, a set of risk–meaningful database constraints, concisely specified in the monitoring procedure.

The level of u is chosen by the system agent, and the greater u is, the more of the risks in the environment can be detected.

The database constraints measured by u are meaningful in such a way that violation of a specific constraint detects the presence of a specific risk. Thus the more database constraints that are specified within the monitoring procedure, the more risks can be detected. Action is taken, on violation of a constraint, and thus on a risk signal, to eliminate the risk detected. The action is execution of a response procedure that does not divert system resources R from normal operation, and thus no slowdown effect.

M(u) is a function of the level of risk–meaningful database constraints specified in the monitoring procedure, and is a measure of the risk detecting effectiveness of the level of constraints u. M(u) is zero for u = 1.0, and increases at a declining rate as u increases, eventually saturating at a value of 1.0. The constant dud–factor parameter d will usually be zero and can be neglected. But with systems where dud risks acted upon lead to throughput capacity losses, such as financial systems [3, 5], d will be above zero, often significantly above.

When d is 1.0, there is no benefit from the monitoring procedure. When d exceeds 1.0, which it can do, especially with financial systems, you are better off to forget about the monitoring approach, for the throughput capacity losses are even greater than with no monitoring procedure.

Complexity measure of a set of risk–meaningful database constraints

The risk–monitoring risk equation above depends on the concepts of *risk–meaningful complexity* and *risk–meaningful complexity measures*. The level of risk–meaningful complexity u in the monitoring procedure increases the rate of detection of risk in advance.

Once the hazard is detected, we can direct a response procedure to respond to it, and so eliminate or reduce the loss–inducing Rr(E) or L term from the basic risk equation:

$$I = KR + G - L$$
$$= RK + Rcr(E) - Rr(E)$$

Risk–meaningful complexity u is the complexity measure of a set of risk–meaningful database constraints. To better understand the concept, observe that for a monitoring procedure operating in real time, the environment may unfold with a degree of unpredictability or randomness. Thus any system operating in real time will be confronted with an unfolding environment that can be characterized by at least one incoming real–time continuous data stream of bits–ones and zeros.

For example, if we have a mobile robot system, there will be a data stream of sensor data from each of the robot's sensors (for example, audio, video and other sensor data). If it is a financial system, there will be a stream of security price and volume data.
For a given data stream, data that has already arrived gives us a past history of the unfolding environment, that is, the historical data string.

Now consider a simple real–time monitoring system, with only one incoming data stream, from a single sensor. If the sensor is appropriate for detecting hazards in the environment, risk in the environment should show up as order or regularity in the historical data string.

If we analyze a typical historical data string, we will be able to come up with a set of bit–string specifications, each of which specifies a regularity that can occur many times in the data string. However, only some of these regularities will be of interest to us, namely those that signal the presence of risk.

Thus what is needed is a specification of those bit–pattern regularities that are meaningful in revealing the presence of risk of throughput capacity loss. Let us call such bit patterns in the historical data string risk–meaningful bit patterns, or rm–bit–strings, or rm–regularities, for purposes of this analysis.

[Note, however, the incoming bit stream, and thus the historical data string, must not be random, that is, the cumulative bit function should not form a random walk, or alternatively, the probability of a bit 1 or bit 0 coming next should not be 50% [2].

For a random walk, the standard deviation of a large set of changes in the cumulative bit–function, each such change measured over a time period of length T, is proportional to T^H, or equals kT^H, where k is a constant. For a random walk, the value of H, the *Hurst constant* for the bit stream, is 0.5. If H = 0.5, no meaningful regularities are possible. Meaningful regularities are possible only for H greater than, or less than 0.5, in which case the cumulative incoming bit stream forms what is called a fractal time series, or a fractal Brownian time function [1].

The cumulative bit stream is simply the sum of the previous bits in the incoming stream of bits. For example, if the bit stream was 1001101010001001, with the most recent bit last, and bit 1 is regarded as +1.0, and 0 is treated as −1.0, the cumulative bit stream value would be −2.0. If in the next period the additional bits are 11001111, the cumulative bit stream value goes to +4.0.

Fractal time series is a large subject, but, although relevant for specific systems, is beyond the scope of this paper. Essentially, if H exceeds 0.5, when the previous bit was a one [or zero], the probability of a one [or zero] coming next in the bit steam will exceed 50%, and there will be a rising or falling trend in the cumulative incoming bit stream. Thus, with H greater than 0.5, if in the previous time period of length T the cumulative bit stream value was up, in the next time period of length T the odds would favor it continuing to be up, and vice versa [1, 2].

In contrast, if H is less than 0.5, there will be reversal prone or counter trend behavior in the cumulative incoming bit stream. In other words, if, in the previous time period of length T, the cumulative bit stream value was up, in the next period the odds favor a reversal and down, and vice versa.]

In order to determine the rm−regularities in a specific environment, a risk investigator must first analyze a very long historical bit string for that environment, so long that even quite rare rm−bit patterns show up sufficiently often to enable the investigator to spot them and to establish that they are correlated with a significant risk, and to enable the investigator to measure that risk.

In general, the more complex something is the longer it takes to describe or specify it. So, suppose we call the rm−bit−strings r1, r2, ? and that, further, we let u be the sum of the lengths of the specifications of all the rm−bit−strings that the monitoring procedure is equipped to detect and initiate a response to. It follows that u must be a measure of the complexity of the risk signals that the monitoring procedure can detect in the environment.
In the risk monitoring procedure, each rm−bit string can be specified as a constraint, for example:

Constraint−23: r23 specification: [e.g 1011001?1010]:
incoming pattern must not be r23;
on violation run proc−23

In other words, if the most recent bit string matching r23 in length is not r23 in content, there is no risk, but there is a risk if there is a content match, since this is a violation of the constraint, causing response procedure proc−23 to execute.

Virus risk−monitoring software, to prevent a computer virus infection, makes use of such constraints in its risk−monitoring procedure or risk−detection component. For every known computer virus, there will be distinctive bit patterns within the virus software. If the virus−protection software detects any of these, it means there is a high risk of a virus being present.

In the case of a single incoming data stream from a sensor, the stored data will form a simple database, and the set of rm−bit−strings will be the core of constraints on that database. Each constraint will be such that an incoming bit string that matches an rm–bit string cannot be stored, without first triggering a constraint violation that signals the presence of risk.

General risk–meaningful database constraints

The above discussion assumes a single data stream from a single sensor that has been selected so as to generate a data stream that will include all relevant risk–meaningful data strings for the system environment. In this simplest case, the occurrence of any of the rm–bit patterns r1, r2, ? would indicate the presence of a risk, requiring execution of one of procedures proc–1, proc–2, ? to avert it. Thus the monitoring procedure would merely have to check for the presence of any of the rm–bit patterns.

Such a simple system is sometimes possible. For example a single sonar sensor in a submarine will detect all relevant data about mines in its path in time for evasive action to be taken.

However, very often such a simple detection system is not possible, because the physical world is usually not simple. Very often risk will go undetected, because of complex or unusual combinations of circumstances.

For example, a sensed incoming bit pattern r3 that normally does not indicate the presence of risk may correspond to a chemical that in most circumstances is safe (e.g. a chlorate salt for use in swimming pools) but which would cause a violent explosion in contact with another chemical with risk–free bit pattern r7 that is also normally safe (e.g. washing–up liquid), and fatal explosions of this kind have occurred–something suicidal terrorists might use to blow up an aircraft. The unavoidable fact is that in a technologically or naturally complex environment it is often difficult for a monitoring system or humans to detect the presence of risk.

In the general case, incoming bit–string data from each sensor has to be checked against a database containing risk–meaningful data about the environment. We call this database *the risk–meaningful database*.

A risk–meaningful database should in principle contain all risk–relevant data about the specific physical environment involved. Much of the data for the risk–meaningful database will be the result of risk–identification investigations into the environment–since the very first step in managing risk is the identification of the specific risks in a given environment.

For detection of risk, a set of constraints for the risk–meaningful database must be constructed, such that, when a sensor or environment monitor, outputs a bit string section already specified as hazardous in the database, one of the constraints will be violated if risk is present. Each constraint is specified as a predicate, which is simply a statement, often quite complex, specifying conditions. Violation of a specific constraint would signal a specific type of risk, and trigger a procedure to deal with it.

In the general case, the constraint predicates may be quite complex and involve many different entities–types from the database, for example:

<database constraint S–37;
<constraint–predicate–37>;
<on violation: run proc–37>

The constraint predicate, constraint–predicate–37, may be a complex specification. For example, in a risk–meaningful database dealing with aircraft and terrain, used for preventing collisions with mountains, the complex specification may involve such entity types as aircraft, mountain, cloud, and so on. Such constraints may be specified in the database system's constraint language [7].

For complex circumstances, the constraint predicates will be complex. Indeed, when video sensor data is involved, and the specific color, shape or motion of an object in the environment, or any combination of these, is what signals presence of risk, the complex constraint required will in most cases be impossible to construct as a conventional database constraint using the database constraint language. Some additional video image processing instructions may well be needed. Still, the resulting specification will be a constraint in the most general sense, and its violation will signal that risk is present.

In the general case, the monitoring procedure thus will need to be equipped with constraints for use with a risk–meaningful database, it being possible that individual constraints will be of great specification length and consequently of great complexity.

3. Relationship between Database Constraint Level and System Throughput

Suppose now that we denote the specification of each of the risk–meaningful database constraints for a given environment by S1, S2, S3 ...SJ? The monitoring procedure will contain a set of conditional (or prescriptive) imperatives of the form:

<constraint–predicate–S1> ; if <S1 violation> then <Alert–1>;
<constraint–predicate–S2> ; if <S2 violation> then <Alert–2>;
. . .
<constraint–predicate–SJ> ; if <SJ violation> then <Alert–J>;
. . .

where each of Alert–1, Alert–2, ... Alert–J ... is a response procedure that can completely eliminate the risk detected by the corresponding rm–constraint violation. On average, although obviously not every time, the execution of a response procedure Alert–J due to constraint SJ violation will give rise to a throughput loss avoidance LJ in I–and possibly, with a financial system especially [3, 5], a new throughput capacity loss dLJ, due to dud risk signals, or false alarms, where d is a constant dud–factor greater than zero.

The monitoring procedure may be said to be saturated with rm−constraints, if all constraints S1, S2, ?SJ,... relevant to the environment being monitored, are specified in the monitoring procedure. If the monitoring procedure for a given environment is unsaturated, that is, not all possible rm−constraints for that environment are coded for in the monitoring procedure, we define the sum of the lengths of S1, S2, S3, ?,SJ, ? as the effective rm−complexity u of the monitoring procedure. Adding additional rm−constraints for that environment to the monitoring procedure will then increase the procedure's effective monitoring complexity, up to its maximum of U.

To derive the relationship between throughput capacity I and rm−complexity u, we note that each rm−constraint SJ violation by the unfolding environment will have a specific frequency of occurrence, as evidenced by the historical data. Some rm−constraint violations will occur very often, whereas others occur only rarely. Consequently, all violations of rm−constraint S23, for example, may give rise to a large throughput capacity loss avoidance L23 on average, per period T, for all executions of Alert−23. In contrast, all violations of rm−constraint S42, for example, may collectively give rise to only a small loss avoidance L42 on average, per period T, for all executions of Alert−42. With financial systems especially, we may also get additional losses, on average, equal to dL23 and dL42, due to responding to dud risk signals, or false alarms.

Note that in this analysis we look backwards over the historical data over many time periods each of length T. An implicit assumption here is that the environment's risk statistics are reasonably stationary, that is, future statistics will be like past statistics, so that the expected behavior in the next future T period is the average for the previous historical set of time periods each of length T.

At this point, we can usefully define a *rm−efficiency coefficient*.

There will be an rm−efficiency coefficient value associated with each of the rm−constraints S1, S2, ?, SJ, ? specified in the monitoring procedure. Using the historical data for risk information, for any rm−constraint SJ, the rm−efficiency coefficient is the contribution to I in terms of average loss LJ eliminated (per time period T) by detection of all violations of SJ (and subsequent executions of Alert−J), divided by the length of the SJ specification.

This means that for a given environment, unfolding over a given period of time, because of the historical statistics, we can order the rm−constraints S1, S2, ? in order of their decreasing rm−efficiency coefficients. Thus the rm−constraint that over the historical period has given rise to the largest contribution to I on average per period T, for the least length of rm−constraint specification or rm−complexity, appears first, say type S1, by virtue of its relative simplicity and high average loss avoidance L1. The one with next largest ratio of contribution to I (say type S2) to length of constraint specification appears next, so that the constraint with the lowest ratio, or rm−efficiency coefficient value, appears last.

We can now define a function M(u) that is a measure of cumulative reduction in I due to responding to the constraint violations, assuming that rm−complexity is added to the monitoring procedure in order of decreasing rm−efficiency coefficient. We call M(u) the risk monitoring effectiveness function.

M(u) has a value between 0 and 1.0. It measures the average throughput capacity loss averted by use of rm−constraint specifications totaling u in the monitoring procedure, as a fraction of the average throughput capacity loss Rr(E) or L, which can be eliminated only by deploying all the possible rm−constraints for the environment, which total U in length. If all possible constraints have been included in the monitoring procedure, this means that all possible hazards can be detected and dealt with. In such a case, u = U, and u cannot be greater. Thus M(U) = 1.0, by definition.

More simply, M(u) is the fraction of the average loss Rr(E) eliminated, due to risk that is averted by means of the rm−complexity level u. So the losses that are eliminated by the level of u are RM(u)r(E), and the system throughput capacity losses, after losses RM(u)r(E) have been eliminated, will be residual losses equal to Rr(E) − RM(u)r(E), that is, R(1−M(u))r(E).

With a few systems, such as financial systems, response action on dud risk signals actually leads to additional system throughput capacity losses [3, 5]. These dud−signal losses approximate some constant times the losses eliminated, and must therefore equal dM(u)Rr(E), where d is a constant dud factor, with a value of zero when dud losses are zero. The dud factor d depends on the chance of a signal being a dud and on the damage a dud does.

Hence, if we have rm−constraint specifications totaling u, since the normal average loss of Rr(E) due to risk will be reduced to Rr(E)[1 − M(u)], average throughput capacity must be:

$$I = R[K + [c - (1 - M(u))]r(E)] - dM(u)Rr(E)$$

which reduces to:

$$I = R[K + [c - (1 - [1 - d]M(u))]r(E)]$$

or, when the dud risk factor d is zero, as will usually be the case:

$$I = R[K + [c - (1 - M(u))]r(E)]$$

where M(u) is zero when u is zero, and where M(u) climbs at a decreasing rate as u increases, until finally as u approaches U and saturation, M(u) approaches 1.0, and saturation.

The function M(u) is a growth function of the general form:

$$G(x) = (1 - e^{-x/k})$$

and saturating at G(x) = 1.0.

In many systems, the dud factor d will be zero, but not with financial systems [3, 5], where d is a measure of the additional system throughput capacity losses experienced due to responding to dud risk signals. When d is zero, there are no such losses. If d is 1.0, the throughput capacity losses due to dud signals equals the system losses L eliminated. If d exceeds 1.0, the losses due to dud signals are greater than the losses L or Rr(E) that the monitoring system is attempting to eliminate. The level of d depends both on the chance of the rm−constraints detecting risks with a low probability of the hazard occurring, and on the extent to which acting on dud signals gives rise to additional throughput capacity losses.

4. Active and Passive Risk−meaningful Databases

It appears that there are two distinct types of risk−meaningful databases, which we refer to as active and passive risk−meaningful databases. In addition, some risk monitoring systems may use many databases, some active and some passive.

An *active risk−meaningful database* is under continual data insertion, in response to one or more data streams coming from a collection of sensors or monitors, some of which may be human. In other words, it will grow in size at a steady rate. A risk−meaningful constraint violation occurs when an insertion violates a constraint. The violation does not hinder insertion of the new data. It does trigger the requisite response procedure, however, since it detects the presence of risk.

A simple example would be a truck inspection database at a sensitive entry point, for example a border crossing. The database contains information about every truck crossing in the past. When a truck arrives, the data pertinent to the truck is entered. Thus truck arrivals generate a stream of incoming data. If the truck has characteristics, which, based on past experience, indicates a risk of some kind, such as possible weapons, smuggling, explosives, and so on, a constraint will be violated, triggering appropriate action on the part of the authorities.

The data for the offending truck will of course be inserted into the database. It may be that, on investigation, the truck is shown to pose no risk, which data would also be entered. If investigation later confirmed that the truck was a hazard, that confirmation data too would be entered, and would also serve to further validate use of the original constraint, and mark the relevant data as indicative of a hazard.

In other cases of active risk−meaningful databases, the data stream might be generated by surveillance sensors, or radar or sonar in military applications, and instead of just one data stream there may be a large number, and the data from all the streams has to be stored. And a risk meaningful constraint may now involve data being inserted from more than one in−coming stream

In contrast, *a passive risk−meaningful database* would not be under continual data insertion, and it would remain approximately constant in size. There would be only occasional updates, to maintain the correspondence between the database and the domain of reality it reflects. The risk meaningful constraints used with the database are of a more general kind, involving not only the data in the database but current data parameters of a kind not stored in the database.

As an example, consider a terrain database stored in an aircraft. The terrain could be the mountainous areas in the western part of North America, in other words, data about mountains, which are hazards to aircraft. The purpose of the database is early detection of the risk of the aircraft flying into a mountain, in times of poor visibility, due to low clouds, fog, or darkness. The data base is passive since there will obviously be no need for it to grow, since the terrain it corresponds to does not grow. There may be some minor need for update from time to time, to reflect significant changes to the terrain. For example, the database would need to be updated to reflect the terrain changes due to the explosion of Mount St. Helen's in Washington State, which removed one whole side of the mountain.

The database would be used as follows. A computer risk monitoring system would monitor the speed, direction and altitude of the aircraft, as well as its latitude and longitude, using satellite global positioning facilities. These parameters, obviously external to the database, would be used in conjunction with the terrain database, to determine if there was a risk of the plane flying into a mountain. If such a risk is determined, the response procedure would alert the crew, probably with a suitable cross−section display of the terrain and the plane's flight path, showing where and when the collision would occur with a mountain if evasive action is not taken. The constraint violation that would determine the presence of risk is thus a constraint of a very general kind, involving not only the terrain data in the database, but other data, like aircraft position, speed, and direction.

Other examples of passive risk meaningful databases would be an database continuing data about all known explosive substances and explosive combinations of substances, a database about all known narcotic substances, a database about all known kinds of firearms and ammunition, a database about firearms registrants, and so on.

Passive databases can obviously be as varied in structure as any of the wide range of databases in existence. Active databases are different, and are more restricted in structure, since they store one or more streams of in−coming data, and associated entity data.

Co−relationships and the structure of active risk−meaningful databases

In the simplest case of an active risk−meaningful database, where there is only a single sensor and a single in−coming data stream, we have data about a sequence of events concerning one or more entities. The in−coming data, that is, event data, will be stored in an event relation, say EVENT_X, in which each tuple has data about an event of type X; each tuple of EVENT_X will have information about the entity instances involved, the time when the event occurred, and one or more attributes, to be updated later, for whether or not any risk alerted by the event turned out later to be a real hazard.

Thus, if an underscore denotes a primary key attribute, the database will have the structure:

ENTITY_A[A, a1, a2, ?]   ENTITY_B[B, b1, b2, ?]   ENTITY_C[ ] ?

EVENT_X[X, A, B, C ?, x1, x2, ?time]

so that there is a one−to−many relationship between ENTITY−A and EVENT_X, another one between ENTITY_B and EVENT_X, and so on. Of course, one or more of the entity relations may be the child of an entity parent, and so on

If we have two sensors and two data streams, we will need two event relations, for example, EVENT_X and EVENT_Y. There may be no relationship between the two types of events, as in:

ENTITY_A[A, a1, a2, ?]   ENTITY_B[B, b1, b2, ?]

ENTITY_F[F, a1, a2, ?]   ENTITY_G[G, b1, b2, ?]

EVENT_X[X, A, B, ?, x1, x2, ?time]   EVENT_Y[Y, F, G, ?, x1, x2, ?time]

However, there may well be a co−relationship [ ] between the two types of events, as in:

ENTITY_A[A, a1, a2, ?]   ENTITY_B[B, b1, b2, ?]   ENTITY_F[F, a1, a2, ?]

EVENT_X[X, A, B, ?, x1, x2, ?time]   EVENT_Y[Y, B, G, ?, x1, x2, ?time]

We can see that EVENT_X and EVENT_Y each have a common attribute B, the foreign key linking B−entities. Since B is not a primary key, in either EVENT_X or EVENT_Y, we have a co−relationship between the two relations.

An even more subtle co−relationship would occur if a futher entity, ENTITY_S, for example, were a parent of both ENTITY_A and ENTITY_F, and so a grandparent of both EVENT_X and EVENT_Y.

Co−relationships are weak relationships [6], ranging in intensity from purely coincidental to reasonably meaningful. It is these weak relationships that are the stuff of intelligence and counter intelligence. For example, suppose EVENT_X lists border crossings by persons (B−entities). Person B6 might occur in EVENT_X many times, since B6 is not a primary key, and will reflect many border crossings by that person. Suppose now that EVENT_Y is a shipment of goods by truck across that border, and that Entity−B is the shipper of the goods.

Suppose now that on a certain day, person B6 crossed the border, as entered in EVENT_X and shipped a truckload of goods across the border at a different border crossing, as entered in EVENT_Y. Is that significant? It might be. It is certainly an aspect of the co−relationship between EVENT_X and EVENT_Y. Of course, it might be pure coincidence. But if past experience showed it to be meaningful then its occurrence should violate a data base constraint signaling the presence of risk. It is in this way that complex constraints involving more than one data stream can detect the presence of risk.

A corollary to this is that co−relationships need to be looked at again, in the light of active risk−meaningful databases. This author investigated co−relationships carefully in the late 1980s [6], but their relevance to risk−meaningful databases was not realized at that time, nor is there any mention of this aspect of co−relationships in the literature. This is a neglect it can obviously pay to remedy, and will be the subject of a later paper.

Summary and conclusions

We have shown how risk−meaningful databases are central to the risk monitoring approach to risk elimination. The risk monitoring approach involves use of a coping procedure consisting of both a risk monitoring procedure for detection of risk, and a response procedure for either eliminating or reducing the risk detected.

The risk monitoring procedure uses violations of risk−meaningful database constraints to signal the presence of risk. The degree to which risk is detected depends on a cumulative measure u of the risk−meaningful constraints. We have shown how there is a need for a function M(u) that is a measure of how effective a specific level of u is in eliminating risk. M(u) is a growth function of u that has value zero when u is zero, that is, when no co risk meaningful constraints are employed, and has a saturation value 1.0, when sufficient constraints are employed to detect all the risk.

M(u) can be used in an extended version of the basic risk equation for systems, to give us the mathematical relationship between throughput capacity and risk, when that risk has been partly reduced by means of a risk coping procedure. When M(u) is 1.0, the risk can be entirely eliminated, assuming that there are facilities in place to allow the response procedures to eliminate any risk detected.

Two types of risk−meaningful databases are identified, namely active rm−databases and passive rm−data bases.

Passive rm−databases are mostly constant in size, and store data about hazardous entities. They can have any of the structures commonly found in databases.

Active rm−databases grow continually, since they store incoming data from monitoring units and sensors. They tend to have a more restricted structure type as a result.

It turns out that the rm−database constraints used for detecting risk with active rm−databases depend heavily on co−relationships. Co−relationships arise from entities, implemented as relations, that have non−primary key attributes in common. They are weak relationships compared with the strong one−to many and many−to−many relationships that normally determine database structure [7]. There is a need for a thorough investigation into co−relationships in a risk−meaningful database context.

References

1. Barnsley, M. F. and others. The Science of Fractal Images, Springer−Verlag, 1988.

2. Beaumont, G.P. Probability and Random Variables, Ellis Horwood Limited, 1986.

3. Bouchoud, J., Potters, M. Theory of Financial risks: From Statistical Physics to Risk Management, Cambridge University Press, 2000.

4. Bradley, J. A risk hypothesis and risk measures for throughput capacity in systems, IEEE Trans. On Systems and Cybernetics, Part A., 33(4), July, 2003.

5. Bradley, J. Elimination of Risk in Systems, Tharsis Books, 2002

6. Bradley, J. Co−relationships, levels of significance, and the source of the connection trap in relational databases, Computer Journal, 35, 1992, pp. A335−342

7. C. J. Date. Introduction to Database Systems, 7th Edition, Addison−Wesley, 2000.

8. Silberschatz, A., Galvin, P. B., Gagne, G. Operating Systems, 6th Edition, Wiley, 2003