A risk-metric framework for enterprise risk management

S. N. Foley
H. Moss

We describe a risk-metric framework that supports enterprise risk management. At the core of the framework is the notion of a risk profile that provides risk measurement for risk elements. By providing a generic template in which metrics can be codified in terms of metric space operators, risk profiles can be used to construct a variety of risk measures for different business contexts. These measures can vary from conventional economic risk calculations to the kinds of metrics that are used by decision support systems, such as those supporting inexact reasoning and that are considered to closely match how humans combine information.

Introduction

Enterprise risk reflects the potential loss that results from failure or uncertainty surrounding enterprise activities [1]. Examples include financial risks due to uncertainty in interest rates and risks due to operational failures in supply chains. Enterprise risk management (ERM) is the process of continuously identifying and addressing these risks [1–3]. Examples of the process are the use of separation of duties to ameliorate the risk of fraud and the use of production planning controls to ensure continuity of a supply chain. This risk management process can be understood in terms of observe, orient, decide, and act—the classic OODA loop [4]—whereby the ability to effectively measure the risks across the enterprise becomes central to an effective risk management process.

One common measure for risk—probability of failure multiplied by resultant loss—is widely used to provide an economic perspective on risk [1]. For example, annualized loss expectancy (ALE) [5, 6] and other economic metrics such as return on investment (ROI) [6] can provide useful indicators that facilitate the risk management process. The ROI calculation may help justify the purchase of a new network server as a means of mitigating a quality-of-service failure. However, risk measurement should not be regarded as an isolated and one-time calculation. The challenge is to closely integrate measurements with the ERM process so that they provide real-time indicators of risk. In the case of the network server example, a challenge might be to track, in real time, how an operational failure in the part of a supply-chain process running on the server has an impact on the current ROI calculation of the server.

Digital Object Identifier: 10.1147/JRD.2010.2043403

Not all risks are quantified in economic terms; for example, balanced scorecards [7] provide noneconomic performance metrics. However, balanced scorecards tend not to be integrated across the ERM process. While the scorecards may provide useful high-level measures for senior executives, it can be difficult for other individuals to reconcile these measures with actions for low-level business operations. Other risk metrics such as those in [8] tend to be domain specific and/or are not necessarily integrated across the ERM process.

ERM frameworks such as those discussed in [2, 3, 9, 10] provide approaches that support the risk management process. These frameworks provide a structure in which to systematically describe the business activities, along with their inherent risks and the controls that are in place to address those risks according to best practice. Using these frameworks can result in a large amount of complex interrelated information, and it can be difficult to trace the relevant threats across the organization or the extent to which risk is mitigated. Typical measures in these frameworks, such as reporting the controls with the highest number of audited failures, tend to be primitive and coarse grained and thus increase the difficulty of effective management of risk across the enterprise.

In this paper, we describe a risk model that allows risk measurement to be closely integrated with the risk management process. The key components of this model are the *risk hierarchy* and *risk profiles*. The risk hierarchy is used to define how risk propagates across a wide range of enterprise risk elements. A risk profile is a container for risk calculations related to risk elements. Unlike conventional risk dashboards that typically build upon a static risk hierarchy limited to fixed risk calculations, risk profiles are

0018-8646/10/$5.00 © 2010 IBM

programmable using a variety of metric space and fuzzy-style logic operators.

This paper is organized as follows. First, we present the core risk model, which is illustrated using a simple risk profile for calculating the annual loss expectancy across the servers of an organization. We do not prescribe particular risk metrics for risk management but provide a programmable framework in which users can build their own metrics. Then, we describe how these risk metrics can be constructed within the proposed model. We also present an example to explore how the model can be used to provide metrics for an ERM framework supporting operational risk. Finally, we outline a prototype implementation of the model in terms of a relational database management system.

Model of risk measurement

Let *Type* denote the set of all *risk element types*, that is, the kinds of enterprise elements with which we wish to associate some degree of risk. Risk types represent anything of interest within an enterprise, including systems, people, processes, tasks, controls, and assets.

Let *Element* denote the set of all *risk elements*, that is, instances of risk types. For example, `webServer` is a risk element of the `Server` risk type. A risk element e is an instance of a risk type $type(e)$; for example, $type(\texttt{webServer}) = \texttt{Server}$. For the sake of clarity, we use a `typewriter` font when writing specific risk `Types` and `elements` and other concrete instances of model variables used as examples.

Risk hierarchy

Risk types are organized relative to each other according to a *risk hierarchy* reflecting how risk elements may aggregate and influence other risk elements. The relationship $depends(C, D)$ means that risk type C *depends* on risk type D. In other words, determining the risk associated with an element of type C is based on the risk associated with an element(s) of type D. For example, $depends(\texttt{Department}, \texttt{Server})$ means that the risk of a department depends upon the risk of its servers.

In addition, a risk dependency relationship $\leftarrow$ is defined between risk elements, where $c \leftarrow d$ means that a determination of the risk associated with c depends on the risk associated with d. For example, $\texttt{sales} \leftarrow \texttt{webServer}$ means that the `sales` department risk is dependent on the risk associated with a `webServer` operating in that department. For the purposes of this paper, we do not consider cross-impact risks; that is, we assume that a dependency between risk elements must be consistent with their type dependencies. In other words, $c \leftarrow d \Rightarrow depends(type(c), type(d))$. Furthermore, we assume that risk dependency is acyclic; that is, a risk type cannot depend on itself under transitive closure ($dependsTransitive()$) of risk dependency.

Risk profiles

A *risk attribute* identifies some risk-relevant characteristic of interest with respect to a risk type. The attribute is intended to reflect a measure of something that is known about the elements of the risk type. For example, a `Server` risk-type has risk attributes `vuln`, `valu`, and `svrALE`, where `vuln` defines the likelihood of compromise to a server with monetary loss (risk attribute) `valu` as a single loss event. Here, the overall risk, calculated as ALE [5, 11], is defined as $\texttt{svrALE} = \texttt{vuln} \times \texttt{valu}$. A risk attribute a is declared as anchored to a unique risk type denoted $own(a)$; for example, $own(\texttt{valu}) = \texttt{Server}$.

While risk attributes define the kinds of risks to be measured, a *risk profile* is a binding from the attributes of a risk type to values for a given risk element. Intuitively, risk profiles provide containers for the calculations associated with a risk element. For example, the likelihood of compromise (attribute `vuln`) of the purchasing server `purServer` has probability (attribute value) 0.2. The set of all risk profile configurations is defined as $Profile \equiv Attribute \rightarrow \Re^+$. Each risk element may have an associated risk profile. The set of all profile configuration states is defined as $State \equiv Element \rightarrow Profile$. Given current state $\sigma \in State$, then $(\sigma\ e)$ is the risk profile of element e, and $(\sigma\ e\ a)$ gives the current value of risk attribute a for element e in state σ.

Primitive and evaluation risk attributes

A risk element e with $own(a) = type(e)$ is considered as explicitly determining a value for this attribute a in its profile. This value $(\sigma\ e\ a)$ in state σ is either externally determined (to the model) via events and data that come from the enterprise or calculated in terms of the values of other attributes in the profile of element e. In the latter case, the attribute a is referred to as an *evaluation* attribute, and $isEval(a)$ is true. Alternatively, in the former case, attribute a is referred to as a *primitive* attribute, and $isEval(a)$ is false. Continuing the server example above, an audit of `webServer` determines values for primitive attributes $(\sigma\ \texttt{webServer valu}) = 1000.0$ and $(\sigma\ \texttt{webServer vuln}) = 0.01$ for current state σ; the evaluation attribute $\texttt{svrALE} = \texttt{vuln} \times \texttt{valu}$ is calculated to be 10.

The values of primitive attributes are assumed to be statistically independent of one another; we do not consider how the value of a primitive attribute might indirectly influence the value of another primitive attribute.

The function $calc(C)$ defines calculation of evaluation attributes based on the profile of an element of risk type C, where $calc : Type \rightarrow Profile \rightarrow Profile$. Given type C and profile p, then $(calc(C\ p)a)$ calculates the risk attribute value for attribute a. We assume that this provides a fixed-point

calculation, that is, $calc(C\ p) = calc(C\ calc(C\ p))$ for any profile p. For example, given $p : Profile$ and attribute a

$$calc(\texttt{workstation}\ p)a \equiv \begin{vmatrix} (p\ \texttt{vuln}) \times (p\ \texttt{valu}) & \text{if } a = \texttt{svrALE} \\ (p\ a) & otherwise \end{vmatrix}.$$

An (nonidentity) attribute calculation may be defined only for evaluation attributes that are anchored to the defining type; that is, for any $C : Type$ and $a : Attribute$, then

$$(\exists p : Profile | calc(C\ p)a \neq (p\ a)) \Rightarrow isEval(a) \wedge own(a) = C.$$

In this paper, attribute calculation is defined as expression assignment; however, in principle, any calculation can be used, for example, a Bayesian network applied to the risk attributes in a profile.

Inherited risk

Risk attribute values for risk elements (in some state) are either determined by reference to the element itself (in the case of primitive and evaluation attributes) or inherited from the risk attribute values of elements in other risk types (inherited attribute). The type–dependency relation determines how these values are inherited. An attribute a is *inherited* in the profile of a risk element of type C if C transitively depends on the type that anchors a, that is, $dependsTransitive(C, own(a))$. For example, the `Department` type inherits the risk attribute `valu` from (anchored to) `Server`; the value of the risk attribute for a specific department is based on an *aggregation* of the values of that attribute for the servers in that department.

Every risk attribute a has an associated aggregation operation $\oplus_a$ (identity 0_a) that defines how its values are aggregated. Given attribute a, then $x \oplus_a y$ is the aggregate of (risk attribute a) values x and y. For example, the values of attribute `valu` are aggregated by numeric addition. Attribute `vuln` values $x, y : [0, 1]$ are aggregated by $x \oplus_{\texttt{vuln}} y$, defined as probabilistic sum $x + y - x \times y$ since attribute values are considered statistically independent. Let prefix operator $\oplus_a A$ denote aggregation over a set A of attribute a values.

Risk events

A *risk event* is any internal or external enterprise event that may influence a primitive risk attribute. Events may be automated or manual, ranging, for example, from real-time results (deriving from sensors or analytics signaling suspect conditions such as intrusion attempts or fraudulent transactions) to scheduled audit procedures designed to test the efficacy of controls. Event behavior is defined by function $\varepsilon : Event \rightarrow Profile \rightarrow Profile$, where $(\varepsilon\ e\ p)$ defines the result of executing event e against profile p. For example, routine auditing checks the server configuration for compliance with the company security policy, including checking for a strong password (e.g., a password that is difficult to guess by both humans and computer programs), disk encryption, and backup service. The audit result is defined by event `ckCompliance.pass` or `ckCompliance.fail` and is applied against the profile of the element under test. For example, a `ckCompliance.fail` updates attribute `vuln` to 0.5; otherwise, it is set to 0.1.

Each event e is anchored to a risk type $own(e)$, reflecting the kinds of element (profiles) that the event may affect. Risk events may affect only the values of primitive attributes that are anchored in the same type as the event, that is, for any event e and attribute a, then

$$(\exists p : Profile | (\varepsilon\ e\ p\ a) \neq (p\ a)) \Rightarrow (own(e) = type(a) \wedge \neg isEval(a)).$$

Risk rollup

The values of primitive risk attributes may change as a result of external events. Such events include ongoing risk events that update attribute values, as well as discovery events that set the initial baseline attribute values. Setting or changing an attribute value in a profile may result in a cascade of changes to the values of other attributes. These cascaded changes result from either evaluated attribute definitions or inherited attribute relationships.

The state $(rollup\ \sigma)$ defines the rollup of state σ, that is, a normalized state σ with all cascading relationships calculated. If a primitive attribute a is anchored to the type of element e, then its value is unchanged in the rolled-up state, that is

$$(type(e) = own(a) \wedge \neg isEval(a)) \Rightarrow ((rollup\ \sigma)\ e\ a) = (\sigma\ e\ a).$$

In rolled-up state $(rollup\ \sigma)$, the following invariant properties should hold.

- The value of an attribute a that is anchored to $type(e)$, in the profile of element e, is calculated using the definition of $calc(own(a))$, that is

$$type(e) = own(a) \Rightarrow ((rollup\ \sigma)\ e\ a) = calc(own(e)(rollup\ \sigma)\ e)a.$$

The value of an inherited attribute a that is not anchored to $type(e)$, in the profile of element e, is calculated as the aggregate of the inherited values of a in the profiles of elements upon

$$dependsTransitive(type(e), own(a)) \wedge type(e) \neq own(a) \Rightarrow ((rollup\ \sigma)\ e\ a) = \otimes_a\{((rollup\ \sigma)\ f\ a)\,|\,f \in Element \wedge e \leftarrow f \wedge dependsTransitive(type(f), own(a))\}.$$

Note that if $type(e)$ does not inherit attribute a, then there is no constraint on the value of a in the profile of e.

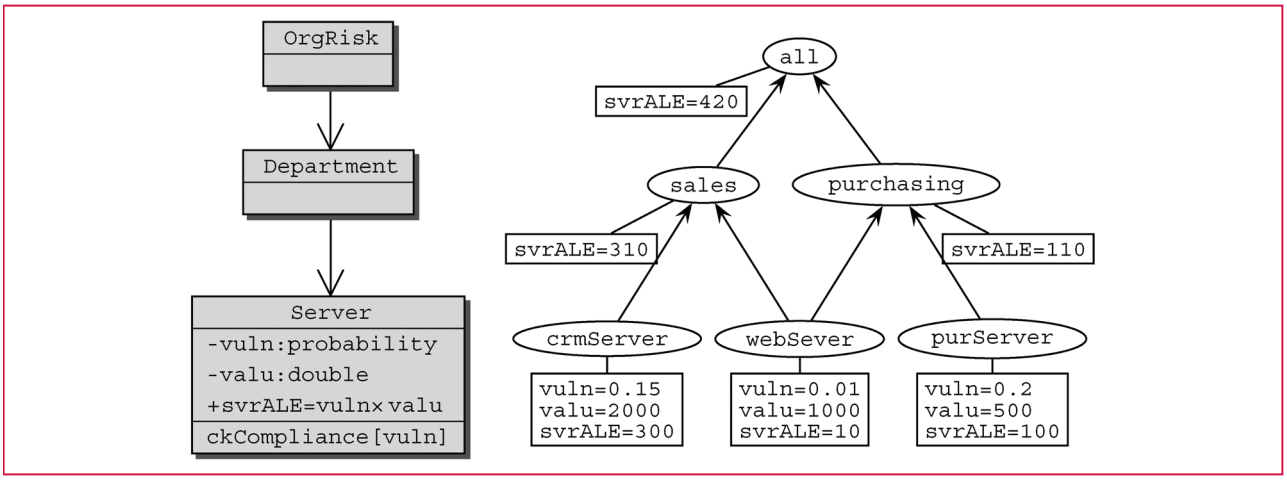

Figure 1

Risk configuration model and rolled-up instance.

Operation (*rollup* σ) is defined as a solution to the equations above. In our current prototype, it is implemented as a relational join across database tables of risk types or profiles.

Example 1

For convenience, we use Unified Modeling Language (UML)-class-style diagrams to construct risk hierarchies, for which risk types correspond to UML classes of risk attributes. **Figure 1** defines a model of a simple risk hierarchy involving servers in departments along with a rolled-up instantiation involving two departments and three servers. An overarching `OrgRisk` class (type) is included, with a single `all` risk element representing the overall organization risk to which all risk rolls up.

Risk attributes are declared within their anchoring risk type (`Server` in this example). Primitive attributes `valu` and `vuln` are declared by indicating their data type (`probability` and `double`, respectively). Evaluated attribute `svrALE` is defined as calculation `vuln` × `valu`. An attribute data type provides default aggregation operators for rollup: `double` provides arithmetic addition as default aggregation; `probability` provides probabilistic sum as a default aggregation operator. An attribute typing system with data-type ordering `probability` ≤ `double` is used to infer that, based on the types in its defining expression, evaluation attribute `svrALE` has data type `double` and uses arithmetic addition for aggregation and rollup. Pass/fail risk events are also included in their anchoring risk type, along with a reference to the primitive attributes upon which they act.

Attributes are declared as private (prefix −) or public (prefix +) in their anchoring type. A public attribute is considered to have an interpretation when rolled up outside of its anchoring type. For example, public attribute `svrALE` provides economic risk associated with each department; risks are aggregated as the sum of individual risks [5]. Sample attribute values (profiles) are provided for each of the servers, along with rolled-up `svrALE` values for the departments.

In the above example, rolling up department `svrALE` risks in order to provide the overall risk for the organization in the profile of element `all` results in a `double` counting of the economic risk of `webServer`. We argue that economic risk such as ALE in this case is not an appropriate metric across risk hierarchies as it "flattens" the hierarchy, resulting in the loss of much of the contextual information about the nature of the risk. In the next section, we consider a more general approach whereby risk is measured in terms a normalized severity of failure.

Measuring risk

The previous section provided a computational model of risk whereby risk calculations are effectively determined by the past (risk) events in the environment. This section considers the kinds of risk metrics that can be used in performing these calculations.

Vulnerability indicators and subjective risk

Risk profiles store measurement data that are used in the calculation of risk. One class of measurement is a *vulnerability indicator*: an independent variable v_i providing a measure of some operational characteristic of the enterprise that is related to the vulnerability. For example, $v_{\texttt{patch}}$ gives the number of days elapsed since a patch (e.g., a fix to a program) was released for a software package but not applied; $v_{\texttt{train}}$ is the percentage of staff lacking up-to-date training.

A vulnerability can be exploited (by an attacker), leading to a *failure*, which is represented as binary dependent

variable ϕ. For example, an intruder has access to a system (via buffer overflow attack), or data loss may occur (due to untrained staff). If failure ϕ is dependent on vulnerability indicators $\upsilon_0, \ldots, \upsilon_n$, then $\Pr[\phi = 1 | \upsilon_0 = i_0, \ldots, \upsilon_n = i_n]$ is the probability of the failure occurring, given indicator values $\upsilon_0 = i_0, \ldots, \upsilon_n = i_n$. In this paper, probabilistic primitive risk attributes implement failure probabilities based on vulnerability indicators that are assumed to be statistically independent. For example, a primitive risk attribute `patch` gives the probability of compromise based on the current value of vulnerability indicator $\upsilon_{\texttt{patch}}$.

A regression analysis on the historical vulnerability indicator values could be used to arrive at the probability distribution. For the purposes of this paper, we assume that this function is monotonically increasing with respect to vulnerability indicator values. Thus, as the value of a vulnerability indicator increases, then the probability of failure (due to the associated vulnerability) increases. For example, we assume that the vulnerability of a system can increase only while it remains unpatched. Exploring nonmonotonic risk within our model is a topic for future research.

Linear regression tends not to be suitable for binary variables, and therefore, we chose logistic regression [12] over a data set of vulnerability indicator υ_i values, obtaining parameter values α and β and estimating the probability of failure as

$$\Pr[\phi_i = 1 | \upsilon_i = x] = \text{logistic}_{\alpha,\beta}(x) = \frac{1}{1 + e^{-(\alpha + \beta x)}}$$

for indicator value υ_i. This provides a compact representation in terms of (α, β) for the distribution that can be associated with the attribute when declared in its anchoring type, and its inverse is easily computed using the logit function [12].

Subjective risk attributes

Part of the motivation in using logistic-based regression analysis is that it provides an intuitive approach for specifying distributions based on the *subjective* knowledge of a domain expert when historical vulnerability data sets are not available. For example, a security administrator advises that there is a low likelihood of compromise to a system that has not been patched for up to ten days; however, there is a high likelihood of compromise if the system remains unpatched after 30 days have passed.

A risk attribute representing vulnerability indicator υ_i can be declared as a subjective probability by specifying a low (lo), high (hi), and residual probability value res, with constraints

$$0 \leq \Pr[\phi_i = 1 | \upsilon_i = lo] \leq res$$
$$1 - res \leq \Pr[\phi_i = 1 | \upsilon_i = hi] \leq 1.$$

Assuming that the probability of failure increases as the value of the vulnerability indicator increases, then we fit these two data points to the curve $\Pr[\phi_i = 1 | \upsilon_i = x] = \text{logistic}(x)$, where

$$\alpha = \left(\frac{lo + hi}{lo - hi}\right) \times \ln\left(\frac{1 - res}{res}\right)$$
$$\beta = \left(\frac{-2\alpha}{lo + hi}\right).$$

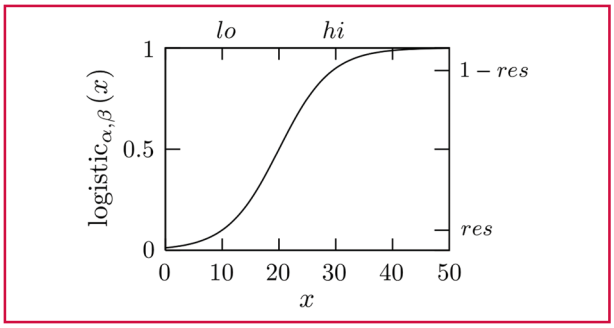

Figure 2

Logistic function with $lo = 10$, $hi = 30$, and $res = 0.1$.

For example, **Figure 2** illustrates the distribution for vulnerability indicator $\upsilon_{\texttt{patch}}$, with $lo = 10$, $hi = 30$, and $res = 0.1$.

Risk events revisited

A risk event serves to update the value of one or more vulnerability indicators, which may increase or decrease the probability of failure. While the risk framework supports any type of risk event, in this paper, we consider the `passfail` and `timer` events that support the execution of conventional audit test procedures.

- A `passfail` event returns a pass (success) or fail result. A fail results in the value of an affected primitive attribute being decremented by 1, while a pass results in an increment by 1.
- A `timer` event returns a pass or fail and updates the affected attribute to the number of days elapsed since the last time it passed.

For example, the daily executed test procedure `ckPatch` fails when the server software is not the most recent version. Defining it as a `timer` event affecting vulnerability indicator `patch` means that, as the value (in days) of $\upsilon_{\texttt{patch}}$ changes, then the probability of failure changes according to the function in Figure 2.

Severity attributes

Risk attributes may be used to provide measures in addition to the probabilities and monetary value. For example, weightings for physical location, the value (nonmonetary) of the data to the organization, how critical the availability of

a web server is to the organization, or the severity or impact of software vulnerability (e.g., [13]) may contribute to evaluating an overall risk score for the data hosted by a server. Since risk attributes may be scored from different ranges of values, it can be difficult to relate the values of different attributes in a meaningful way. *Severity* provides a way to interpret the meaning of a risk attribute value and to normalize disparate attribute values to a common severity scale.

Severity is defined as a measure of the relative significance or impact of a risk attribute. A risk attribute a has a severity generator $\Gamma_a = \mathit{Attribute} \rightarrow \Re \rightarrow [0,1]$ and a (raw) value v of attribute a is interpreted as severity value $\Gamma_a(v)$. Here, the notation $[0,1]$ indicates a range of values from 0 to 1. Values 0 and 1 represent the lowest and highest severity values, respectively. It is assumed that severity is monotonic, that is, given raw values v_1 and v_2 for attribute a, then $v_1 \leq v_2 \Leftrightarrow \Gamma_a(v_1) \leq \Gamma_a(v_2)$. The severity generator is a logistic regression of subjective knowledge from a domain expert specified in terms of *lo* and *hi* value bounds.

As an example, an attribute `valu` with severity range [500, 1,500] indicates that a server valued less than $500 is considered to have little value, while a server valued over $1,500 has a high value. Severity can also be used to normalize an evaluation risk attribute in order to control risk tolerance, for example, mapping `svrALE` (Example 1) to the severity range $[200, 400]$ means that an annual loss expectancy of less than 200 is tolerated.

Aggregation and risk metrics

Severity values may be interpreted in terms of fuzzy logic [14], whereby the logistic function approximates the fuzzy set with two points. Triangular norms are operations that generalize the fuzzy logic operators, and a variety of t-norm/t-conorm operators [15, 16] can be defined. For example, the product fuzzy logic has a probabilistic product (t-norm) and sum (t-conorm). The latter provides a useful default aggregation operator for rollup. We adopt a simple attribute data-typing system based on ordering $\texttt{probability} \leq \texttt{severity} \leq \texttt{double}$; this is used to determine a type for an evaluation attribute based on the data types of the attributes in its defining calculation and also to provide a default aggregation operator for the rollup of the attribute.

Example 2

The `Server` risk profile from Example 1 is redefined as follows:

$$\texttt{valu} : \texttt{severity}[\texttt{lo} = 500, \texttt{hi} = 1500]$$
$$\texttt{patch} : \texttt{probability}[\texttt{lo} = 10, \texttt{hi} = 30]$$
$$\texttt{svRisk} = \texttt{patch} \times \texttt{valu}$$
$$\texttt{timer ckPatch}[\texttt{patch}].$$

In this case, the subjective range $[\texttt{lo} = 10, \texttt{hi} = 30]$ is used when calculating the probability of failure of a server risk element based on the value of vulnerability indicator `patch` that is updated by the `timer` procedure test `ckPatch`. Attribute `svRisk` is calculated using the normalized values of attributes `valu` and `patch`.

Attribute `svRisk` provides a metric indicating the risk of server failure, the severity (impact) of which can be measured at different points (risk elements) in the organization risk hierarchy. The types `probability` and `severity` use the t-conorm probabilistic sum (disjunction) as the default aggregation operator with the result that `svRisk` is also rolled up using a probabilistic sum. Given the risk hierarchy instance in Figure 1, with the revised `Server` risk profile above, and if $\texttt{svRisk}(e)$ represents the (rolled-up) value of attribute `svRisk` in the profile of risk element e, then the rollup of `svRisk` to risk element `all` is calculated as

$$\begin{aligned}
&\texttt{svRisk}(\texttt{all}) \\
&\quad = \texttt{svRisk}(\texttt{sales}) \oplus_{\texttt{svRisk}} \texttt{svRisk}(\texttt{purchasing}) \\
&\quad = \texttt{svRisk}(\texttt{crmServer}) \oplus_{\texttt{svRisk}} \texttt{svRisk}(\texttt{webServer}) \\
&\qquad \oplus_{\texttt{svRisk}} \texttt{svRisk}(\texttt{webServer}) \\
&\qquad \oplus_{\texttt{svRisk}} \texttt{svRisk}(\texttt{purServer}).
\end{aligned}$$

The failure of `webServer` has an impact on two departments and is therefore considered to have a double impact on the overall (`all`) organization risk measure.

A variety of t-norms provide useful severity aggregation operators and can also be used in the calculation of evaluation attributes. For example, the use of t-conorm $\max(x, y)$ as the aggregation (rollup) operator for `svRisk` in the previous example provides a simple fuzzy risk metric [15, 16]. The risk model provides a programmable framework in which a range of metrics can be specified and calculated in a hierarchical context.

A goal in selecting the aggregation operators to be used in a risk profile is to match more closely how humans aggregate information. Zimmerman and Zysno [17] discovered that when making decisions, humans do not necessarily aggregate according to the linearity of a t-norm; that is, there may be potential for nonlinearity in the way that combinations are perceived. For example, a human may place proportionally greater significance on the aggregation of low-severity items rather than on moderate-severity items. Our prototype currently supports the compensation aggregation operator for neutral element $n : [0,1]$ [18, 19]. Intuitively, this uni-norm operator may be thought of as a combination of the probabilistic product when operand severity values are less than n and the probabilistic sum when operand severity values are greater than n. Using this operator, for example, with $n = 0.2$, aggregating `svRisk` causes the rollup to be less sensitive to the aggregation of low risk values.

Enterprise risk management

The previous sections illustrate the risk model by considering risk in terms of simple organizational elements. In practice,

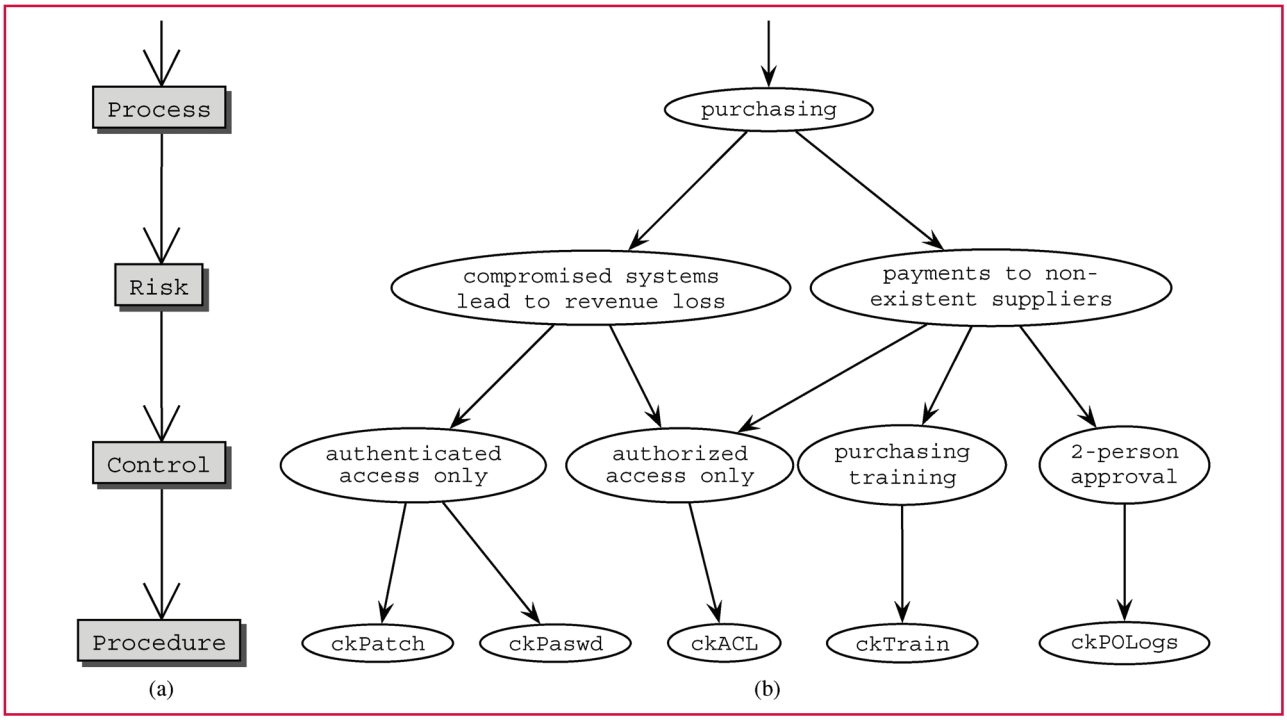

Figure 3

Hierarchy of risk types. (a) Abbreviated risk hierarchy. (b) Control fragment.

the risk model can be used to define risk in terms of any elements that can be organized as a risk hierarchy. In this section, we outline how the risk model can be closely integrated with the elements of an ERM framework.

Figure 3 depicts a simplified ERM framework as a hierarchy of risk types. A `Process` represents the business activities within an enterprise and can be associated with one or more risks. `Risk` is the uncertainty in a process that could have an adverse impact on the business policy. A `Control` is a framework for the management of an activity or set of activities meant to prevent a business process risk from occurring. A `Procedure` is a set of test activities, executed at a predetermined frequency to ensure that a control is effectively working as design. The figure provides a partial instantiation of this hierarchy for the `purchasing` process.

Two risks are identified that impact the objectives of the `purchasing` process, and controls are introduced in order to mitigate those risks. The risk that revenue loss can occur as a result of illegal access is mitigated in part by the following two controls. First, users on the system must be authenticated. Passwords are used to authenticate authorized users, and periodic checks (`ckPasswd`) for weak passwords are carried out in order to prevent password-guessing attacks. An intruder might obtain unauthenticated access to the system by exploiting software vulnerability, and ensuring that the software is patched and up to date helps mitigate this risk. Frequent audits of installed software versions (procedure test `ckPatch`) validate the effectiveness of this control. Second, only users who require access as part of their job function should be authorized to use the purchasing system. Frequent audits (`ckACL`) of the system access control policy validate the effectiveness of this control.

The risk of making payments to nonexistent and/or unapproved suppliers is mitigated in part by the following three controls. First, only authorized users should have access to the payment system. Second, the staff should be given ongoing training with monthly audits (`ckTrain`) of staff completion. Third, every purchase order must be approved by at least two individuals, with random checks (`ckPOLogs`) for compliance of the logs.

Figure 4 provides a UML-class-style diagram of the simplified ERM risk hierarchy that incorporates a risk profile and attribute declarations. Risk attributes are anchored in the `Control` type and provide measures of the effectiveness of the control at mitigating a risk. These control risk profiles can, in turn, be "rolled up" the risk hierarchy, providing measures of the effectiveness of risk management in the context of the identified `Risk`, the parent `Process`, and so forth.

Recall that a risk element may have a number of associated risk profiles and that, in this example, an individual control can have different profiles that are used to manage measurements of its effectiveness with respect to servers,

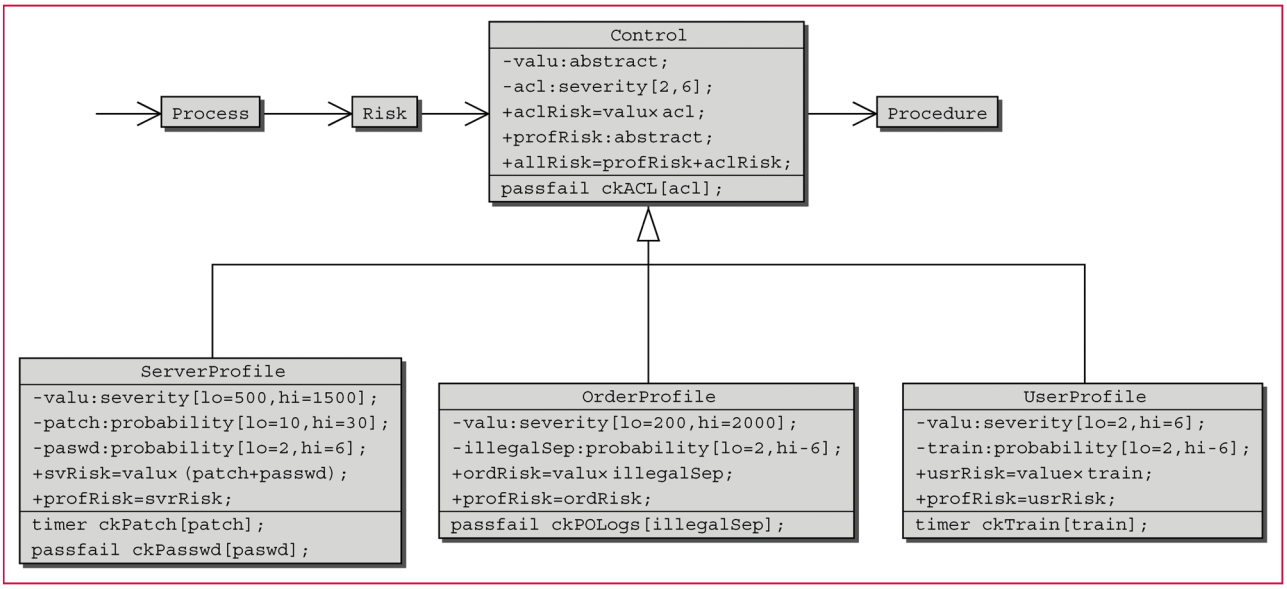

Figure 4

Risk profiles for ERM.

purchase orders, and users. This is represented using a UML-class generalization in order to define subclasses (types `ServerProfile`, `OrderProfile`, and `UserProfile`) of the `Control` profile superclass. The following risk attributes are defined, providing information about the assets or vulnerability indicators.

- `valu` defines a severity measure of the value of the asset. It is defined as abstract, as it has different severity interpretations depending on whether it is the value of a server, order, or user. For illustration, the server and order values are based on the monetary value, while a user `valu` is some measure of the importance of the role of the assets in the organization.
- `acl` is the likelihood of failure as a consequence of an invalid access control policy. It is updated via `passfail` procedure test `ckACL`: a value of two or fewer failures is considered low risk. When `acl` has a higher value, then the (stateful) semantics of `passfail` require a series of successful procedure tests before it can be considered as having a low chance of failure.
- `aclRisk` defines the risk of access control policy noncompliance and can be calculated for servers (server policy noncompliance), users (risk of users having noncompliant access), and orders (risk of noncompliant access to orders).
- `allRisk` defines the overall risk and, for a given profile, is the (default probabilistic) sum of the `aclRisk` and any other profile-specific risks `profRisk` of the subclass.
- `svrRisk` gives the server-specific risk based on `patch` and the likelihood of failure as a consequence of a user having a weak password (attribute `paswd`).
- `ordRisk` gives the order-specific risk based on the likelihood of an illegal order as a consequence of a violation of the two-person order rule (attribute `illegalSep`).
- `usrRisk` defines the user-specific risk based on the likelihood of compromise as a consequence of the staff not taking training or not maintaining training (attribute `train`).

Risk profiles provide measures on the effectiveness of a control, whereby effectiveness is determined by the outcome of the procedure test. For example, the risk profiles for the control `authenticated access` only provide scores on the outcome of procedure tests for each server (for example, `webServer` and `purServer`), for users, and for orders. The aggregation of profiles associated with the control provides measures on its effectiveness, including an overall measure `allRisk`. When rolled up to the risk "`compromised systems lead to revenue loss`," it provides a measure of the effectiveness at mitigating the risk. This provides a multilevel approach to presenting and investigating risk. The higher the risk attribute in the risk hierarchy, the more abstract the measurement. For example, a high `allRisk` score on the `purchasing` gives a general indication of the health of that business process. This might prompt a drill-down in the hierarchy to identify the poorly mitigated risk(s), with a further drill-down to discover the noncompliant user(s) with a high `usrRisk` score. Here, we use the term drill-down in an information technology

sense to indicate, for example, the traversal of a risk hierarchy in search of specific information.

For the sake of clear exposition, the examples in this paper are intentionally simple. In practice, internal control catalogs can have a large number of risks and complex controls, and it is beyond the scope of this paper to consider how these catalogs might be encoded in terms of the proposed risk model. In one experiment using an existing catalog, we built a preliminary risk profile whereby each risk in the catalogue corresponded to a unique risk attribute, updated by the procedure tests of the controls of the risk. For example, a risk attribute corresponding to "`compromised systems lead to revenue loss`" (from Figure 3) is updated (`passfail`) by `ckPatch`, `ckPasswd`, and `ckACL`. This is preliminary work, and further research is needed to evaluate how the risk model might be used in a practical environment.

Prototype

A relational database management system provides a practical approach to managing the risk profiles. Risk types map to the database schema, with schema attributes implementing risk attributes, and the risk hierarchy is implemented via schema (key) dependencies. Risk events (test procedures) are implemented as database triggers that, when executed, update the related primitive attributes and also perform any calculations associated with evaluation attributes.

In performing a rollup, the model computes a join over tables corresponding to an instance of the risk hierarchy with attributes as defined by the risk attributes. However, Structured Query Language does not provide arbitrary aggregation operators over attributes; only primitive arithmetic aggregation operators are directly available, for example, `SUM`, `AVG`, and `MAX`. The current risk model prototype restricts the permitted aggregation operators to *Archimedean* t-norm/conorms [20], which means that attribute aggregation is simply implemented by `SUM` over its column in the database table.

A high-level risk-specification language has been implemented that is comparable to the UML diagrams used to illustrate the examples in this paper. This language is used to specify arbitrary risk hierarchies, risk attributes and their types (including types `probability`, `severity`, and `double`), and risk events (including `passfail` and `timer`). A compiler translates a specification in this risk language to a DB2* database implementation with associated triggers and stored procedure queries for rollup. The generated implementation model can be integrated with an existing ERM framework and thus provides risk profile support.

Discussion and conclusion

The risk model presented in the section "Model of risk measurement" provides a user-programmable framework in which to monitor and measure the effectiveness of controls at mitigating risk. While measurements are made at the level of individual controls, they can be aggregated and then used to calculate more general risk measures from the perspective of different contexts or levels in the organization. The examples in the sections "Model of risk measurement" and "Measuring risk" illustrate *organizational-centric* measurements, whereby organizational structures provide the risk hierarchy and the contexts from which to present the risk. Organizational-centric structuring is typical for security risk management dashboards such as those discussed in [10] and [21]. The *ERM-centric* measurement used in the example in the section "ERM" provides multilevel and contextual risk measurements that are closely integrated with ERM framework elements and therefore provide a basis for informed decision making.

The section "Measuring risk" considered risk metric construction within the risk model. Rather than limiting the measurement to common metrics such as the number of control failures or top-failing controls, risk profiles are programmable, using a variety of metric space and fuzzy-style logic operators. Encoding the security metrics described in [8] within the risk model is an interesting topic for future research. While [21] supports fuzzy (security) risk metrics, it does not consider other forms of aggregation nor their use in an ERM context. Risk calculations in the risk model can be based on historical behavior or subjective information based on the knowledge of domain experts. For example, scores from the Common Vulnerabilities and Exposures database [13] can be used as subjective weightings when calculating the risk of system/software failure.

A risk profile is a user-programmable container for risk calculations related to risk elements. While the examples in this paper have focused on operational (security) risk, the risk profiles can also be used to manage calculations related to other risk scenarios. For example, an organization manufactures five key products, and continuity requires that no single part is secured from a single vendor, that no part is utilized across more than three products, and that no single part inventory is less than 60% of the sales pipeline. A product risk profile can be used to coordinate this calculation whereby if the part becomes unavailable, then the organization would either source an additional vendor or initiate a greater inventory. Using the risk model to manage calculations for other risk classes is a topic for future research.

Acknowledgments

This research was carried out while Simon Foley was a member of Corporate Security Strategy, IBM, on leave of absence from University College Cork, Cork, Ireland. He would like to thank IBM and Stuart McIrvine for making this possible.

References

1. *Risk Management—Vocabulary—Guidelines for Use in Standards*, ISO/IEC Guide 73:2002, 2002.
2. *Enterprise Risk Management-Integrated Framework*, Committee of Sponsoring Organizations of the Treadway Commission (COSO), Jersey City, NJ, 2004.
3. C. Abrams, J. von Känel, S. Müller, B. Pfitzmann, and S. Ruschka-Taylor, "Optimized enterprise risk management," *IBM Syst. J.*, vol. 46, no. 2, pp. 219–234, Apr. 2007.
4. J. Boyd, *Patterns of Conflict,* Available as presentation. [Online]. Available: www.d-n-i.net/fcs/ppt/boyds_ooda_loop.ptt
5. *Special Publication Risk Management Guide for Information Technology Systems (800-30)*, U.S. Government Printing Office, Washington, DC, 2002.
6. A. A. Groppelli and E. Nikbakht, *Barron's Finance,* 4th ed. Hauppauge, NY: Barron's Educational Series, Inc., 2000.
7. R. S. Kaplan and D. P. Norton, *The Strategy-Focused Organization: How Balanced Scorecard Companies Thrive in the New Business Environment*. Boston, MA: Harvard Business School Press, 2001.
8. A. Jaquith, *Security Metrics: Replacing Fear Uncertainty and Doubt*. Reading, MA: Addison-Wesley, 2007.
9. S. J. Root, *Beyond COSO: Internal Control to Enhance Corporate Governance*. Hoboken, NJ: Wiley, 1998.
10. G. Evans and S. Benton, "The BT risk cockpit—A visual approach to ORM," *BT Technol. J.*, vol. 25, no. 1–13, pp. 88–100, Jan. 2007.
11. L. A Gordon and M. P. Loeb, "The economics of information security investment," *ACM Trans. Inf. Sys. Security*, vol. 5, no. 4, pp. 438–457, Nov. 2002.
12. D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*. Hoboken, NJ: Wiley, 2000.
13. *CVE International. Common Vulnerabilities and Exposures.* [Online]. Available: cve.mitre.org/
14. L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Commun. ACM*, vol. 37, no. 3, pp. 77–84, Mar. 1994.
15. B. Schweizer and A. Sklar, *Probabilistic Metric Spaces*. New York: North Holland, 1983.
16. D. Dubois and H. Prade, "A review of fuzzy sets aggregation connectives," *Inf. Sci.*, vol. 36, no. 1/2, pp. 85–121, 1985.
17. H.-J. Zimmermann and P. Zysno, "Latent connectives in human decision making," *Fuzzy Sets Syst.*, vol. 4, no. 1, pp. 37–51, Jul. 1980.
18. B. Buchanan and E. Shortliff, *Ruled Based Expert Systems, The MYCIN Experiment of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley, 1984.
19. E. P. Klement, R. Mesiar, and E. Pap, "On the relationship of associative compensatory operators to triangular norms and conorms," *Int. J. Uncertainty, Fuzziness Knowl. Based Syst.*, vol. 4, no. 2, pp. 129–144, 1996.
20. H. T. Nguyen, V. Kreinovich, and P. Wojciechowski, "Strict Archimedean t-norms and t-conorms as universal approximators," *Int. J. Approx. Reason.*, vol. 18, no. 3/4, pp. 239–249, Apr./May 1997.
21. M. Dondo, "A fuzzy risk calculations approach for a network vulnerability ranking system," Defense R&D Canada, Ottawa, ON, Canada, Tech. Rep. Memorandum 2007-090, 2007.

Received March 16, 2009; accepted for publication June 1, 2009

Simon N. Foley *University College Cork (UCC), Cork, Ireland (s.foley@cs.ucc.ie)*. Dr. Foley earned a Ph.D. degree from UCC in 1988. He is a Statutory Lecturer in computer science with University College Cork, Cork, Ireland, where he teaches and carries out research on computer security. He has more than 70 international peer-reviewed publications on security, and his research interests include security modeling, distributed access controls, risk management, and security psychology. He serves on the editorial board of the *Journal of Computer Security* and the *Journal of Privacy, Security, and Integrity* and has served as the Program Chair of the IEEE Computer Security Foundations Workshop and the ACM/ACSAC New Security Paradigms Workshop.

Harold Moss *IBM Software Group, Cambridge, MA 02142 USA (hmoss@us.ibm.com)*. Mr. Moss is an Emerging Technologies Architect in the IBM Corporate Security Strategy Team. In that role, he is responsible for providing technical insights for existing and emerging security technology directions. He also had responsibility for verifying and validating architectural direction in a number of solutions based on cloud computing and Web 2.0 technologies, to ensure alignment with customer needs and other IBM assets. Currently, he is a member of several IBM architecture boards and champions the delivery of assets for cloud computing and Web 2.0 technologies.