Software Risk Management and Insurance

Orna Raz
Institute for Software Research, International
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213 USA
+1 412 268 1120
orna.raz@cs.cmu.edu

Mary Shaw
Institute for Software Research, International
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213 USA
+1 412 268 2589
mary.shaw@cs.cmu.edu

ABSTRACT
How can we promote reuse of code, data and services? How can we make it easier to combine on-line resources to perform specific tasks? One serious impediment is the risk of relying on software that you do not control, especially the difficulty of determining whether the software is dependable enough for the specific task at hand. We concentrate on one form of economic risk mitigation, insurance, and explore its suitability for the software domain we are interested in. After reviewing the basic principles of insurance we present some feasible directions for dealing with software related issues and raise some software engineering research challenges.

1 INTRODUCTION

Software reuse has many potential benefits, yet people often prefer to develop their own solutions. When the reuse community discusses impediments to reuse (e.g. [18], [20]) one of the underlying themes is various vulnerabilities to software that is not completely under your control.

"Reuse" usually refers to reuse of code, where the main risks are related to interfaces, side-effects and implicit assumptions. But you can also reuse data and services. We concentrate on an instance of this more general form of reuse: on-line resources combined for a user-specific task, which we call open resource coalitions [17]. Risks related to reuse here are even more severe than in regular code reuse. The resources remain under control of their proprietors, who are often not aware that you are using their resources, let alone aware of the specific ways you do so. As a result, the resources may change or cease to exist at any time. How much should you trust such software? How can you determine its fitness for your task? A mismatch between your trip destination and your hotel reservations will cost you money, time and inconvenience. However, if you make strategic business decisions based on erroneous information you might lose your business. If we could find ways to estimate and manage such risks, this development model would be more attractive to more people, especially non-sophisticated users who would otherwise be unable to create such software.

Risk management consists of risk assessment and risk control [7]. Risk assessment addresses whether the software you intend to use is good enough for the task at hand. We concentrate on risk control since we are interested in evaluating the usefulness of the insurance model. In risk control a major distinction can be made between prevention and mitigation. The common practice in computer science is to make sure the software works properly by performing some combination of verification and validation. This is an example of prevention or reduction of risk. We believe this can rarely be done cost-effectively for the everyday software we are interested in. We discussed some of the problems involved in [17]. Here we investigate mitigation. Mitigation has both technical and economic aspects. Fault tolerance is a form of technical mitigation. Its main concern is technical recovery — trying to bring the system back to service by containing the damage done. But what about compensation for damages? Insurance is a form of economic mitigation. It provides a way to manage risk by sharing and accepting the transfer of risks.

Can insurance provide techniques for managing software risks? To examine this question we first review the basic principles of insurance. Then we look at what is needed to enable insuring software and suggest ways to advance in this direction. Finally, we indicate some related software engineering research possibilities.

2 INSURANCE

Despite all precautions, we cannot completely avoid risks. Insurance provides a means of reducing financial loss due to the consequences of risks, by spreading or pooling the risks over a large number of people. Insurers offer their product (the insurance policy) to buyers (policy-holders) for a price [13]. Actuarial science consists of building and analyzing mathematical models to describe the process by which money flows into and out of an insurance system [12]. Risk theory provides effective tools for doing so. Following [12], we concentrate on cash outflow due to claims payments.

To understand whether insurance can be applied to software we first need to understand the basic principles of insurance, the risk models used and the data these models require. We begin by reviewing risk principles, following [13]. We then follow [9] and distinguish long-term vs. short-term insurance. What matters most in short-term insurance is whether claims occur — the number of claims and their magnitude. In long-term insurance, such as life insurance, the dominant concern is the time lapse until a claim. The main issues are therefore related to the way the economy will behave during this time (e.g. interest rate and inflation). We concentrate on short-term insurance since its time frame is the more appropriate model for software.

Risk Theory

An *actuarial risk* is a phenomenon with economic consequences that is subject to uncertainty with respect to one or more actuarial risk variables: occurrence, timing and severity. An insurer views risk as the probable amount of loss resulting from claims. A *loss event* or a *claim* is an accident in which an insured suffers damages which are potentially covered by their insurance contract. The *loss* is the dollar amount of damage suffered by a policy-holder as a result of a loss event (may be zero). The *amount paid* is the actual dollar amount paid to the policy-holder as a result of a loss event (may be zero). The *severity* can be either the loss or amount paid random variable. (The above are from [12]).

Risk management is targeted at coping with risks. Possible strategies are:

- Avoidance (do not go bungee jumping any more)
- Prevention/reduction (exercise).
- Personal assumption (pay health related costs yourself)
- Sharing (join a group health plan)
- Transfer (retired persons — transfer some health care costs to government programs)

Sharing and transfer can be accomplished through insurance. In addition, insurance companies often work with their clients to promote avoidance and prevention or reduction of risk. *Insurance* aims to reduce financial loss by spreading the risk over a group of people. The group needs to be as homogeneous as possible regarding risk characteristics, yet large enough to share the risk. This way people who did not experience a loss help to repay the losses of the few who did. The process of finding these risk groups is called risk classification. It enables determination of coverage and its price for each group. *Coverage* describes both the specific protection (bounded car repair) and the kind of insurance held (automobile). By defining the specific protection an insurer tries to minimize the degree of insurable risk.

Not all risks are insurable. For a risk to be *insurable* the loss event related to it must have the following characteristics:

- The event is accidental or unintentional
- The event can only result in financial loss, never in gain
- The loss is significant
- The amount of loss can be determined
- The insurer is capable of assuming the risk (there can be loss events, such as war, that comply with all the previous conditions but are uninsurable because of the huge loss)

To distinguish insurable from non-insurable risks it helps to categorize risks into pure risks and speculative risks. *Pure risks* are accidental and unintentional events that have a probability of resulting in financial loss but never in financial gain. Pure risks are generally considered insurable. They include three main categories: personal, property and liability risks. In *speculative risks*, such as gambling, there exists a probability for both financial loss and gain. This probability is highly unpredictable, therefore it is not possible to insure against such risks.

To be able to purchase insurance a person must have an *insurable interest* in the person or object to be insured, meaning they would suffer a genuine loss should the event occur.

The compensation in case of a loss is set according to the *indemnity principle*: the policy-holder should be fairly compensated for the loss but should not profit it.

Short-Term Insurance Risk Models

A term of up to one year is considered short. Examples include most health, property and liability insurance systems. Claims payment plays a dominant role in the risk assumed in a short-term insurance contract. To construct a risk model we need distributions for two basic random variables: the number of insurance claims in one period and the loss amount, given that a loss event has occurred [13].

- The *frequency distribution* is the distribution of the number of claims in one period. When the portfolio of risks is considered as a whole this process is usually modeled as a Poisson process [5]. A short review of short-term insurance models with emphasis on frequency of claims distributions can be found in [15].
- The *loss distribution* is the probability distribution associated with either the loss or the amount paid [12]. Often highly skewed, heavy tailed distributions are used to model loss represented as average payment per claim (heavier tail: more probability is pushed into higher values). [11] presents a short review of loss distributions.

Modeling the frequency and loss distributions enables determination of the minimal amount that must be charged to policy-holders in a risk group in order to cover claims generated by this risk group. This amount is called *pure* or *net premium* and is defined as the rate of claims times the loss [4]. The *actual premium* charges are determined based on the pure premium of each risk group along with factors related to the business environment, such as the cost structure of the insurance company and marketing [4]. We concentrate on pure premium.

The estimate for the pure premium is based on data which is inherently incomplete and may also come from different sources. We follow [12] in the presentation of this issue. Risk classes are not completely homogeneous. For example, part of the difference is due to random variations in the underlying claim experience and part is due to the policy-holder being indeed a better or worse risk than the class at large. Two main sources of data are available: internal and external. *Internal data* is historical data of a particular group or individual. *External data* is historical data from similar policies. *Credibility* is a procedure by which external and internal data are combined to better estimate the expected loss (or any other statistical quantity) for each policy-holder. Often this is simply a weighted average of external and internal data. [10] presents an introduction to credibility theory.

3 APPLYING INSURANCE TO SOFTWARE

Our interest is in software reuse risks. These are risks that arise when the online code, data or services you are using misbehave and as a result cause you an insurable loss. We are interested in using insurance to alleviate the consequences of such failures, thus enabling broader reuse. The risks we are interested in arise due to the characteristics of software.There are existing forms of insurance for risks related to the business setting, mainly risks related to a traditional business going on-line. These include insurance against potential lawsuits involving fraud, libel or invasion of privacy and against risks related to external attacks such as theft, tampering and destruction of information resources [19] (also includes a list of companies offering such insurance). For example, PayPal [2] offers automatic free insurance (by Travelers) for up to $100,000 against unauthorized withdrawals from your on-line account or any of your checking accounts accessible from it. Insurance, such as "errors & omissions" insurance, is also available for IT solution providers [16].

It seems attractive to apply insurance to the software economic risk mitigation problem. This long-existing solution is well understood, and there is a large body of theory and experience we can build on. But can the theory be mapped to our domain? We begin by considering how insurance applies to the software domain. We suggest a plausible model for software insurance and concentrate on anomaly detection, which we view as a first step towards such insurance.

Mapping Insurance Models to Software

Fundamental principles of insurance models need to be re-examined for software. We want to provide solutions for the realistic situation in which rather than having a fully validated and verified resource we have a resource that will fail in certain ways and people that rely on the resource in different ways. When a resource fails, different people will incur different consequences, ranging from no loss to high loss. We begin by examining some possibly problematic issues, where our domain seems to differ from traditional insurance. These are related to the data needed to construct predictive models, the kind of models appropriate to this data, and what is an insurable risk. We suggest plausible approaches to handle each, in our domain of open resource coalitions.

Data

Risk modeling requires data on risks. The data needs to be both of sufficient quality and quantity, to provide a sound statistical basis for predictions. Quality is related to the source of the data. Sufficient quantity depends on the methods used.

Traditional insurance uses historical claims data to model the probability density function (pdf) of the frequency of claims and of loss. These models are then used as a basis for predictions. Usually a large body of external data (from similar policies) is available, but the internal data (from a specific policy in question) is more sparse. Credibility theory is then used to merge the two sources. Since actuaries use parametric models they only need moderate amounts of data.

For software, even if historical data of claims were available, it would become obsolete due to the rapid software turnaround. Instead of historical claim data we suggest collecting current usage results and concentrating on failures. We assume failure data is a sufficient surrogate for claim data. This needs to be verified. Using failure data, risk models for a given enumeration of resources can be based on:

- Failure frequency distribution. This is analogous to the claims frequency distribution.
- Failure consequences distribution. This is analogous to the loss distribution.

To enable risk modeling we need to address two major issues. One is figuring out the general form of these distributions. This is the topic of the next section (risk models). The other is getting data for a specific resource version. We discuss this topic here.

Data needs to be collected and analyzed fast enough to enable tracking versions. Due to the demand for rapid data collection we may need to simulate actual usage. A challenge here is to find appropriate simulation models and use cases.

Data can be collected by "black-box" or "white-box" approaches. In a black-box approach the data is collected by a neutral third party, using publicly available versions of the resources. In a white-box approach, the third party is a partner of the producer. The third party has an opportunity to collect data in advance (before a release) and to take advantage of additional kinds of data (e.g. the source code, development process). Underwriters Laboratories, for example, provides product certification using a white-box approach, including certification of software components as part of an end-product and a software component recognition program (standards and conformance testing) [3].

Setting a premium involves extrapolating from the actual or simulated failure information. However, there are limits to extrapolation. The time frame for which extrapolation can produce good results greatly depends on the characteristics of the underlying process producing the data. A periodic process enables reasonable predictions for any future period based on data of a single period. For example, web page accesses of most weeks can be predicted based on data of one week. If no periodicity is identified, reasonable extrapolation will probably (at best) be relevant for future time units that are somehow proportional to the time units in which the data was collected. Technically, we can collect data as frequently as communication allows, so we can have as much data as we need. Depending on the underlying process, however, we may be limited by the need to collect data over a large enough period of time (which may be dynamically refined).

Risk models
Two major classes of estimators are parametric and non-parametric. In parametric estimation you assume the distribution belongs to a known parametric family (e.g. Normal) and use the data to guide the selection of the form of the distribution and to calibrate its parameters (e.g. μ, σ) In non-parametric estimation the true distribution of the observations is not assumed to belong to a known parametric family, and the distribution is represented solely by the empirical distribution. There are costs and benefits to each. If the underlying distribution is known, parametric estimators are best. Relatively little data can provide good estimations. Most importantly, inferences can be made beyond the population that supported the model. In case of doubt regarding the form of the distribution, it is better (from the standpoint of accuracy) to use non-parametric methods. Such methods require large amounts of data for good results.

Actuaries use parametric estimators to predict short-term insurance costs [12], with a large collection of distributions to choose from.

What estimators are appropriate for our domain? We have little if any prior knowledge about the underlying distributions. We do not know whether the kind of models actuaries use are appropriate for our data. We hope to find a subset of actuarial parametric models that is appropriate, and to benefit from the strict statistical rigor of traditional actuarial estimations. If we cannot do that, we intend to use non-parametric estimators. We expect the statistical rigor of such estimations to suffice for providing a better alternative to the current personal assumption of risks, as we hope to enable pooling risks. In either case, we expect the precision of our estimates to improve as we get more experience and data. COCOMO [6] provides examples of empirical estimators related to software. Software reliability engineering [14] provides examples of parametric estimators. It seems feasible to use the latter as guidance for distributions that may be appropriate for software failure data.

Insurable risks
As we have seen in section 2, a loss event must have certain characteristics for the risk related to it to be insurable.

First, we need to scope the coverage provided by the insurance. We need ways to define which risks are potentially covered by our insurance, as well as ways to assign responsibility for failures. Both of these are currently open research problems. Ideally, we want to insure only against those risks that can be connected to what is specified about the software. Unfortunately, such specifications rarely exist. Examples of the kinds of risks we want to consider include software problems resulting in: failure to supply any data, failure to update the daily forecast of a weather resource. An example of a risk beyond the scope of our insurance is problems in the weather model itself.

Candidate loss events need to be clearly defined and then examined against the following characteristics.

- *The event is accidental or unintentional.*
  The event can only result in financial loss.
  Assuming the resources are provided in good faith and given that you use then for your own ends, risks seem to have the characteristics of pure risks.
- *The loss is significant.* Our domain seems appropriate. We want to make open resource coalitions appropriate for more demanding use, through insurance. If we can put instruments in place for risk management, we hope the boundaries of appropriate usage will expand and so will the opportunities of reuse (even though it will remain inappropriate when catastrophic consequences are possible). For example, we can imagine the combination of fault tolerance and insurance for risk control. As your coalition becomes more dependable, you trust it to perform tasks requiring higher assurance. This means it is now more usable to you, but if an unexpected failure occurs the consequences are likely to be more severe. Proper insurance can alleviate such risks.
- *The amount of loss can be determined.* In our domain, loss is due to the consequences of software failures of online resources. We need means for analyzing such consequences. Determining loss is related to coverage. The coverage has to be related to the anticipated loss and provide an upper bound on the amount paid due to a loss event. To determine loss it may be useful to look at two levels of coverage. The first is limited coverage for direct costs of using the resource. The second is higher coverage for consequential damages of relying on the resource. It might be prudent to begin with smaller, direct damages coverage and extend to higher, consequential damages coverage as we gain more experience and more data. Examples of direct damages include the cost of: using the resource, using an alternative resource, recovering from backup and user time. To determine consequential damages we need to look at each case separately. Imagine a couple, from different continents, planning their wedding, using online resources. They inspect, for each of their home towns, costs and possibilities related to organizing a party and airfare. They notice airfare is extremely cheap for flights originating at the bride's-to-be home town. Only after making commitments based on this fact do they discover

the airfare resource they were using was producing erroneous data due to some internal problems. The consequences for the young couple are quite severe: in order for their families to be able to attend the wedding they now have to change their plans and break previous commitments. Apart from time and hardship, this costs them a lot of money. Now imagine a small pesticide business. They apply pesticide based on online weather conditions posted the same day. If no rain is expected, they rent a plane and apply the pesticide. If it rains after they apply the pesticide, they need to do it again. The profitability of the business relies heavily on the weather resource. Any problems at the resource that go unnoticed, such as not updating the weather conditions posted or updating for the wrong location, might result in financial loss and in customer dissatisfaction. Looking at these examples, it seems we can provide general guidance regarding how to turn user-specific loss determination into a problem that is more similar to the traditional insurance situation. For example: look at decisions based on a resource and quantify the cost of a decision due to misleading information.

- *The insurer is capable of assuming the risk.* Open resource coalitions should not be used when the consequences of a failure can be catastrophic. This limits the possible loss. The loss is also limited by the scope of our suggested insurance, by what we consider as relevant loss events. In addition, in traditional insurance, the insurer expects to get an improved estimation of expected total losses by pooling risks [12]. In our domain, pooling seems possible. For resources provided over the Internet there is a large enough community using then to share the risk. Recall that our setting involves widely used resources, not unique software (such as spacecraft software). Insurance of unique risks is also possible, as done by Lloyd's [1], for example, but it presents different problems. Once we have some data we can begin to address the issue of whether such pooling can improve the estimate for expected total losses.

Form of Software Insurance

To summarize, we see many similarities between traditional insurance and insurance against software reuse risks, as well as potential and promise for the latter. The key issue for software insurance seems to be the data — the need to replace historical claims data. Other differences follow from that. We need to find appropriate models for the distributions of failure frequency and consequences. Statistical analysis of software risk modeling may be less tight, therefore the fiscal control lower. Yet, we believe it will enable to provide an enhanced ability to manage reuse related risks. This will achieve our goal of qualitatively improving users sense of control and lowering their perceived risk.

Anomaly Detection

So far we have discussed what the form of software insurance would be, how to get from data to numbers that enable decision making. But how do we get the data in the first place? We need both to recognize loss events and to determine loss. We concentrate here on recognizing loss events. Consequences of loss event are beyond the scope of our work here. In most traditional insurance settings, humans recognize loss events and file claims. Relying on humans in our setting is not feasible. Automation changes the problem of noticing loss events and allocating responsibility. It may also change the way claims are filed to automatic (or semi-automatic) filing that is coupled with loss event detection.

Loss events in our domain are software failures. We classify such failures into communication failures (cannot get data), syntax and format failures (cannot parse the data) and semantic failures (data doesn't make sense). Failure detection techniques exist mainly for the first kind and for some parts of the second. We therefore concentrate on detection of semantic failures. These are failures to supply the intended semantic due to software failures. Examples include failure to provide timely updates of near real-time data, such as current weather conditions, changing data that should be static, and neglecting to provide complete information, such as missing an important news item about a requested topic.

To model the frequency of failures you can look at the frequency of anomalies after determining which indicate failure with high probability. A first step is to be able to detect anomalies.

We are studying semantic anomaly detection. Our raw data consists of HTML or XML pages from resources. Each page is the result of a specific operation on the resource, such as querying a weather resource for Pittsburgh's forecast. The data may contain any available combination of results of operations (identical operations or not) on resources (a single resource, multiple similar resources such as weather servers or multiple distinct resources that are combined in a coalition). We deal with real-world resources so our data is noisy. To avoid the need for excessive user and domain specific information and to tolerate noisy data, statistical techniques are appropriate. It seems feasible we can collect enough data to enable statistical inference. This is especially attractive as a way to enable anomaly detection while avoiding the thorny issue of precise specifications.

We anticipate, for example, detecting problems related to timeliness, consistency or completeness of near real-time data resources using existing techniques such as linear combinations of values from multiple resources using a moving window [21] and dynamic de-facto invariant detection [8] (again using a moving window). The details are beyond the scope of this paper, but preliminary experiments, using invariant detection on stock quote data, look promising. After making a lot of simplifying assumptions we were able to detect inconsistencies and missing data.

Data mining techniques are designed to deal with large quantities of noisy data. These characteristics are excellent match for our data. There may, however, be some mismatches, as suggested by work done at IBM [4], which used data mining for discovering insurance risks in traditional property and casualty insurance.

4 RESEARCH QUESTIONS

We have presented issues related to applying insurance to software risk mitigation and suggested plausible directions for dealing with some of them. Here we list some related software engineering research questions.

- Defining risks and failures. What are the types of insurable risks related to an open resource coalition (ORC)? How do we define and identify failures?
- Assigning responsibility for failures.
- Refining the ORC model with real data.
- Distributions for failure frequency and consequences.
- Semantic anomaly detection. What does semantic anomaly mean for different types of resources? Can we generalize beyond a specific domain and user?
- Data mining and semantic anomaly detection. Are data mining techniques applicable? What techniques are appropriate for different types of anomalies?
- Combining insurance and fault tolerance as risk mitigation techniques. What is the relationship between fault tolerance and loss? (e.g. adding fault tolerance makes severe failures less frequent but also makes people trust the system more, resulting in higher consequences of failures). Can insurance estimates of failures guide cost-effective fault-tolerance?
- Generalizing software insurance to more settings. We examined what is needed to enable insurance for ORCs. We suspect this can be generalized to other software classes as well. What are these classes? What changes to the data and models will be needed?
- Simulation models. Can we find representative use cases to enable both fast data collection and good predictions?
- Loss estimation. How can we get from software failures to their effect the user?

5 ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under Grant CCR-0086003.

The authors thank Philip Koopman and his students for their helpful comments and suggestions.

REFERENCES

[1] Lloyd's home page. http://www.lloyds.com.

[2] Paypal home page. http://www.paypal.com.

[3] Underwriters laboratories. http://www.ul.com/pscs.

[4] C. Apte, E. Grossman, E. Pednault, B. Rosen, F. Tipu, and B. White. Probabilistic estimation based data mining for discovering insurance risks. Research report, IBM T. J. Watson Research Center, 1999.

[5] R. E. Beard, T. Pentikainen, and E. Pesonen. *Risk Theory: The Stochastic Basis of Insurance*. Chapman and Hall, third edition, 1984.

[6] B. W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, 1981.

[7] B. W. Boehm. Software risk management: Principles and practices. *IEEE Software*, pages 32–41, January 1991.

[8] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support program evolution. ICSE 99, pages 213–224, 1999.

[9] J. C. Hickman. Introduction to actuarial mathematics. In *Actuarial Mathematics*, volume 35 of *Proceedings of Symposia in Applied Mathematics*, pages 1–3. 1986.

[10] P. M. Kahn. Overview of credibility theory. In *Actuarial Mathematics*, volume 35 of *Proceedings of Symposia in Applied Mathematics*, pages 57–66. 1986.

[11] S. A. Klugman. Loss distributions. In *Actuarial Mathematics*, volume 35 of *Proceedings of Symposia in Applied Mathematics*, pages 31–55. 1986.

[12] S. A. Klugman, H. H. Panjer, and G. E. Willmot. *Loss Models: From Data to Decisions*. John Wiley & Sons, 1998.

[13] T. Lowe. *The Business of Insurance: A Comprehensive Introduction to Insurance*. Health Insurance Association of America, 1998. (Primary author).

[14] M. R. Lyu. *Software Reliability Engineering*. IEEE Computer Society Press, 1995. (Editor).

[15] H. H. Panjer. Models in risk theory. In *Actuarial Mathematics*, volume 35 of *Proceedings of Symposia in Applied Mathematics*, pages 17–30. 1986.

[16] D. Raikow, M. Mehler, and B. Napach. Watch your step. *SmartPartner*, February 2001.

[17] O. Raz and M. Shaw. An approach to preserving sufficient correctness in open resource coalitions. Tenth International Workshop on Software Specification and Design (IWSSD-10). IEEE Computer Society, November 2000.

[18] B. Stroustrup. Language-technical aspects of reuse. Forth International Conference on Software Reuse. IEEE Computer Society, April 1996.

[19] J. Voas. The cold realities of software insurance. *IT Pro*, January — February 1999.

[20] J. Waldo. Code reuse, distributed systems, and language centric design. Fifth International Conference on Software Reuse. IEEE Computer Society, June 1998.

[21] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for coevolving time sequences. ICDE 2000, San Diego, CA, Feb 2000.