Early Dealing with Evolving Risks in Long-Life Evolving Software Systems*

Le Minh Sang Tran

University of Trento, Italy
tran@disi.unitn.it

Abstract. Existing risk assessment methods often rely on a context of a target software system at a particular point in time. Such contexts of long-living software systems tend to evolve over time. Consequently, risks might also evolve. Therefore, in order to deal with evolving risks, decision makers need to select an appropriate risk countermeasure alternative that is more resilient to evolution than others. To facilitate such decision, we propose a pioneer method taking the uncertainty of evolutions and outputs of a risk assessment to produce additional information about the evolution resilience of countermeasure alternatives.

Keywords. risk assessment, evolving risk, max belief, deferral belief.

1 Introduction

Long-life software systems keep evolving to continuously satisfy changing business needs, new regulations, or the introduction of new technologies. Such evolutions might expose the software systems to new risks, and might make the output of the current risk analysis on the software systems become partially obsoleted. Consequently, the software systems might be no longer secure.

Risk assessment methods compatible with ISO 31000:2009 typically rely on a context of the target software system at a particular point of time. These methods help to identify risks and countermeasure alternatives (*i.e.* set of countermeasures). Decision makers then need to select the most appropriate alternative to implement to mitigate unacceptable risks. However, when the context evolves, risks might also evolve. Previously acceptable risks might become unacceptable or vice versa, or new risks emerge [1, Chap. 15]. For example, any risk mitigated by SHA-0 based countermeasure was acceptable before 2004, but might be unacceptable later since SHA-0 was efficiently attacked[1]. Thus, a current countermeasure alternative may no longer be appropriate and it is necessary to develop new ones to address the evolving or newly emerging risks. Obviously, implementing new ones to replace for obsoleted ones may be more expensive than having a one that still be appropriate. Decision makers then need to take into account the evolution of risks while selecting countermeasure alternative.

* This work is supported by the European Commission under the project EU-FP7-NoE-NESSOS

[1] http://en.wikipedia.org/wiki/SHA-0#SHA-0, site visited on March, 2013

While there are lots of risk assessment methods *e.g.,* [1,3–6,9], few supports a systematic selection on risk countermeasure alternatives [4, 6, 9], and even less mentioning evolving risks [2]. Traditional risk assessment methods typically perform on a context at a particular point of time and hence could not guarantee the risk assessment results in an evolving context. Concerning to evolutions, in [2], the authors proposed a general technique and guideline for managing risk in changing systems. However, they did not mention the uncertainty of evolutions (*i.e.* likelihoods of occurrences), and how to use these information to support the decision making process.

The approach in this paper is an effort to fill some of that void. The focus of this paper is not on how to obtain the uncertainty information, but rather on how to make use of them to produce additional factors to support the decision making process. In particular, we propose a method to take the potential changes, their uncertainty, and risk assessment outputs to quantitatively evaluate the evolution-resilience of a countermeasure alternative.

2 Terminology

- *Context*: includes all required information to do risk assessment such as assumptions about the working environment, requirements model, targets to be protected and so on. It is the premises for and the background of the risk analysis, as well as the purposes of the analysis and to whom the risk analysis is addressed [1, Chap. 5]. According to ISO 31000:2009, a context includes all external factors (*e.g.,* regulatory, environment) and internal factors (*e.g.,* business process, policies, standards, system functions, reference models).
- *Before context*: is the current context at the current time.
- *After context*: is the future context with potential changes.

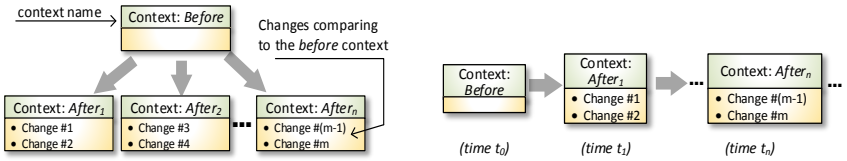
3 The Proposed Method

Our proposed method includes following steps:

- Step 1 *Identify evolving contexts:* identify the current context and all its possible evolutions.
- Step 2 *Perform risk assessment:* do risk assessment on identified contexts.
- Step 3 *Model context evolution:* describe the context evolution model based on identified contexts and their corresponding risk assessment outputs.
- Step 4 *Perform evolution analysis:* perform analysis on the context evolution model to calculate the quantitative evolution metrics to support the selection of an appropriate countermeasure alternative to address risks.

3.1 Step 1 – Identify Evolving Contexts

This step takes all documents about the planned and potential changes of the system as inputs. We consider four different evolution perspectives: *maintenance,*



(a) Before-after evolution (b) Continuous evolution

Fig. 1. The evolution perspectives of contexts

before-after, *continuous evolution*, and *hybrid evolution*. The first three perspectives were discussed in [1, Chap. 15]. The *maintenance* perspective relates to the outdate of a risk document of an existing system. Hence it is not the focus of this work. The *before-after* perspective predicts future contexts by anticipating planned and unplanned changes in the current context. The *continuous* evolution perspective predicts the evolution of the current context over time based on planned gradual changes. The *hybrid* evolution perspective is the combination of *before-after* and *continuous* ones where both planned and unplanned changes are considered in a sequence of time.

We abuse the notation of *before* and *after* contexts to represent these evolution perspectives (except the *maintenance* one). Fig. 1(a) demonstrates the *before-after* evolution perspective. A context is depicted as a rectangle with child compartments. The first compartment shows the context name, and the second compartments exhibits the changes comparing to the *before* context. In this perspective, a *before* context might have many possibilities to evolve to other *after* contexts, denoted as *evolution possibility*. At the end of the day, exact one possibility materializes. Each evolution possibility associates with an *evolution probability* which is the likelihood that a possibility materializes. This uncertainty is because "the only certainty is that nothing is certain"(*Pliny the Elder*[2]). Fig. 1(b) illustrates the *continuous* evolution perspective where changes happen continuously. The *before* context at current time t_0 might evolve an *after* context at time t_1 which might continuously evolve at time t_2, and so forth.

After contexts can be identified by using any input document that describes potential changes (either planned or unplanned) in the current context. Unplanned changes could be anticipated by domain experts by using several techniques such as brainstorming with chalk and blackboard, or techniques for requirements changes anticipation. Readers are referred to [10, Chap. 6] for a more detailed discussion of these techniques. The evolution probabilities are the experts' belief that evolution possibilities might happen. The probability semantic is accounted by using the game-theoretic approach described in [8].

3.2 Step 2 – Perform Risk Assessment

In this step, we employ a state-of-the-art risk assessment method (*e.g.,* Attack Trees [5], Cause-Consequence Diagrams [3], and CORAS [1]) to perform risk

[2] Gaius Plinius Secundus (23 – 79), a Roman naturalist, and natural philosopher.

assessment for identified contexts. The outcome of this step is list of risk countermeasure alternatives, which are also the output of a risk assessment method.

A *risk countermeasure alternative* includes a list of countermeasures, and the residual risks (with residual risk level) of a system after implementing the countermeasures. A countermeasure could be a security controls (*e.g.,* technology, policy), or a high level security requirement that mitigates risks. A risk level is a pair of the likelihood by which a risk might occur, and its impact. Based on risk level, a risk is categorized, such as *acceptable* or *unacceptable*. A residual risk level is the risk level after implementing countermeasures.

When performing risk assessment on *after* contexts, we can do either a full risk assessment from scratch, or an incremental risk assessment taking advantage on the risk assessment on the *before* context. Needless to say, the former strategy does not use resources efficiently. The latter is better since it only addresses the changed parts of the *after* context comparing to the *before* context [1, Chap.15].

3.3 Step 3 – Model Context Evolution

This step takes the identified contexts and their corresponding risk countermeasure alternatives to establish the context evolution model. We employ the approach described in [8] to model the context evolution in terms of evolution rules. There are two kinds of rules: *observable rule* and *controllable rule*. The former captures the way how the context evolves. The latter captures different alternatives to address risks in each context. An evolved context, as aforementioned, is foreseen with a certain evolution probability. To the sake of simplicity, we assume that the evolving contexts identified in Step 1 are complete and mutual exclusive. In other words, exact one of the *after* contexts materializes at the end.

Let $\mathcal{C}$ be an context, and $\mathcal{C}_i$ be the i^{th} *after* context of $\mathcal{C}$, and CA_j be a risk countermeasure alternative of $\mathcal{C}$. The observable rule (r_o) and controllable rules (r_c) are described as follows.

$$r_o(\mathcal{C}) = \left\{ \mathcal{C} \xrightarrow{p_i} \mathcal{C}_i \,\middle|\, \sum_{i=1}^{n} p_i = 1 \right\} \tag{1}$$

$$r_c(\mathcal{C}) = \{ \mathcal{C} \rightarrow CA_j \,|j = 1..m \} \tag{2}$$

where n is the number of *after* contexts of $\mathcal{C}$; p_i is the evolution probability for which $\mathcal{C}$ evolves to $\mathcal{C}_i$; m is the number of risk countermeasure alternatives of $\mathcal{C}$. The sum of all p_i is 1 since the *after* contexts are complete and mutual exclusive.

The *before-after* evolution perspective is represented by an observable rule. The *continuous* and *hybrid* evolution perspectives are represented as a sequence of observable rules where the current context of an observable rule is the *after* context of another observable rule, so on and so forth.

Fig. 2 shows a graphical visualization of the context evolution model of the *hybrid* evolution perspective. The observable rule is denoted by connections from a *before* context to *after* contexts. The decorators on the connections are the evolution probabilities. To denote the controllable rule, the rectangles representing

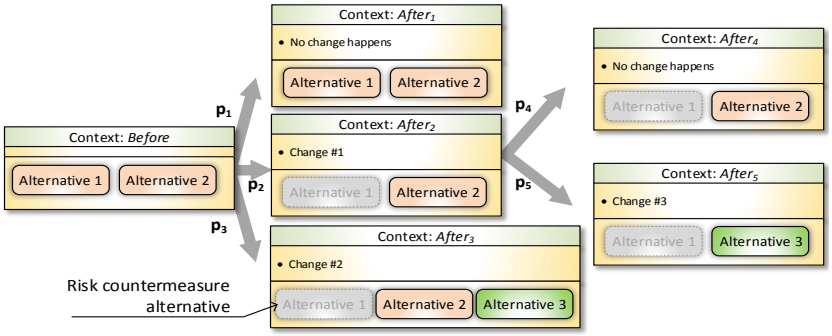


Fig. 2. The context evolution model.

context are extended with a new compartment containing risk countermeasure alternatives which are represented by round rectangles. The controllable rule then is understood as different risk countermeasure alternatives of a context.

3.4 Step 4 – Perform Evolution Analysis

This step performs an evolution analysis on the context evolution model to calculate evolution metrics that support the decision making process. In particular, the evolution metrics aim to answer the question that to what extent a risk countermeasure alternative can resist the evolution. This analysis relies on two quantitative metrics: *max belief* and *deferral belief* [8][3].

Max Belief (*MaxB*): is the maximum belief that a risk countermeasure alternative will be appropriate if evolutions happen. By term *appropriate*, we mean the residual risks after applying the countermeasure alternative in the evolved contexts will still be acceptable. So, the system will still be safe.

Deferral Belief (*DefB*): is the belief that a risk countermeasure alternative will be inappropriate after evolutions happen. It is also the belief by which the implementation of the risk countermeasure alternative should be delayed until the context is clearly known.

We define a binary function appropriate() that takes two inputs: a context $\mathcal{C}$, and a risk countermeasure alternative CA, to produce 1 if CA is appropriate within $\mathcal{C}$, or 0 otherwise. The *max belief* and *deferral belief* of CA for the *before-after* evolution of the context $\mathcal{C}$ are as follows.

$$MaxB(CA|_{\mathcal{C}}) = \max_{\{\langle \mathcal{C} \xrightarrow{p_i} \mathcal{C}_i \rangle \in r_o(\mathcal{C}) | \mathsf{appropriate}(\mathcal{C}_i, CA)\}} p_i \tag{3}$$

$$DefB(CA|_{\mathcal{C}}) = 1 - \sum_{\{\langle \mathcal{C} \xrightarrow{p_i} \mathcal{C}_i \rangle \in r_o(\mathcal{C}) | \mathsf{appropriate}(\mathcal{C}_i, CA)\}} p_i \tag{4}$$

To the perspective of evolution-resilience, a better alternative is one that has a higher *max belief* and a lower *deferral belief*.

[3] We rename the *Residual Risk* metric in [8] to *deferral belief* to avoid naming conflict.

4 Conclusion and Future Work

The context in long-life evolving software systems might evolve over time. As the result, the risks of software systems might also evolve. Under the evolution of risks, the systems might be no longer secure. Therefore, new countermeasures need to be implemented to mitigate new risks. This however is much more expensive than addressing these risks at development time.

We are aware of only one study in the field [2] that introduced general techniques and guidelines for managing risk in evolving systems, but did not mention to the uncertainty of evolutions. Our work is pioneer in filling in the gap by proposing a method inspired from metrics in [8] to evaluate the evolution-resilience of the risk countermeasure alternatives which are the outputs of risk assessment. This provides more insights about the evolving risks to support decision makers in selecting the most appropriate alternative. We refer interested readers to [7] for a more detail discussion.

In future, we plan to extend the proposed method to provide quantitative metrics combining the *max belief*, and the *deferral belief* with the risk assessment output to facilitate the selection of countermeasure alternative. A promising approach is to employ the *Overall Cost* metric described in [9]. Also as a part of future work, we aim to develop a technique exploiting these quantitative metrics to suggest the countermeasure alternative that efficiently address the evolving risks with (or without) an optimal cost.

References

1. M. S. Lund, B. Solhaug, and K. Stølen. *Model-Driven Risk Analysis. The CORAS Approach.* Springer, 2011.
2. M. S. Lund, B. Solhaug, and K. Stølen. Risk Analysis of Changing and Evolving Systems Using CORAS. In *FOSAD*, 2011.
3. S. Mannan and F. Lees. *Lees' Loss Prevention in the Process Industries*, volume 1. Butterworth-Heinemann, 3rd edition, 2005.
4. T. L. Norman. *Risk Analysis and Security Countermeasure Selection.* CRC Press, Taylor & Francis Group, 2010.
5. B. Schneier. Attack Trees: Modeling Security Threats. *Dr. Dobbs Journal of Software Tools*, 24 (12):21–29.
6. G. Stoneburner, A. Goguen, and A. Feringa. Risk Management Guide for Information Technology Systems. Tech. report, NIST, 2002.
7. L. M. S. Tran. Early Dealing with Evolving Risk in Long-life Evolving Software Systems. Tech. report, University of Trento, 2013. http://disi.unitn.it/~tran.
8. L. M. S. Tran and F. Massacci. Dealing with Known Unknowns: Towards a Game-theoretic Foundation for Software Requirements Evolution. In *CAiSE'11*, 2011.
9. L. M. S. Tran, B. Solhaug, and K. Stølen. An Approach to Select Cost-effective Risk Countermeasures Exemplified in CORAS. *CoRR*, 2013. http://arxiv.org/abs/1302.4689.
10. A. van Lamsweerde. *Requirements Engineering - From System Goals to UML Models to Software Specifications.* Wiley, 2009.