Modelling Assumptions and Requirements in the Context of Project Risk

Andriy Miranskyy[a], Nazim Madhavji[b], Matt Davison[c], and Mark Reesor[d]
[acd]Department of Applied Mathematics, University of Western Ontario, Canada
[b]Department of Computer Science University of Western Ontario, Canada
{[a]amiransk, [c]mdavison, [d]mreesor}@uwo.ca, [b]madhavji@csd.uwo.ca

1 Introduction

Many researchers have emphasized the importance of documenting assumptions (`As`) underlying software requirements (`Rs`) [2]. However, `As` and `Rs` can change with time for reasons such as: (i) an `A` or `R` was elicited incorrectly and subsequently needs to be changed; (ii) operational domain changes induce changes in the `A` and `R` sets; and (iii) the change in validity of an `A`, or desirability of an `R`, respectively, causes the validity of another `A` or desirability of an `R` to change.

In practical terms, the invalidity of `As` or `Rs` is a source of problems [2] for software developers and users alike. For an operational system, invalidity[1] of the `As` or `Rs` may imply anything from a diminishing value of the existing software system to a software failure. For a software project in the planning or development stages, such invalidity translates into what we call *invalidity* risk. The risk is that the software being developed may not be as desirable upon completion as first imagined. It is thus important to be able to predict, during the early stages of requirements engineering and periodically from then on during development, the amount of invalidity risk inherent in the software project. Note that in a software project there are other important risks to content with. For the quantification of invalidity risk, there is a need to model the relations amongst `As` and `Rs` and, using these relations, to compute a measure of invalidity risk. In Section 2, we describe our model and how it works. To put such a model into practice, we need to consider at least two scenarios. One is intra-release cycle-time, where invalidity risk is predicted at the start of the project for times between the inception and completion of the project. This would give us intra-release risk trends. The second scenario is prediction over multiple releases. This would give us a risk trend over a longer period of time. The full paper (see [1] at `http://www.apmaths.uwo.ca/asl/papers/tr645.pdf`) describes an algorithm to cover both of these scenarios and gives an example (from a banking application) of how the model could apply in practice. Here, we consider only the first scenario due to limitation of space.

[1]For lack of space here, we ignore the case of an invalid assumption or a requirement becoming valid though our model can handle this situation.

2 Model description

There are four key aspects to our model: One, there exist interactions between `As` and `Rs`. Three general types of interaction are `A`↔`A`, `R`↔`R`, and `A`↔`R`, as described in the full paper.

Two, let $V_{(\cdot)}(j, t)$ be a binary variable, which is set to 1 when the j-th `A` (resp. `R`) in set $(\cdot)$ is valid (resp. desirable) at time t and is 0 otherwise. The interaction between `As` and `Rs` is modeled (through the development cycle) using a Boolean Network (BN) framework [1]. This describes the connections between `As` and `Rs` as well as the factors which cause them to change state.

Three, the system's invalidity risk is modeled as a time sequence of system states where a system state is defined by the validity and desirability of the total set of `As` and `Rs` at any given time. Specifically, during system usage, the operational domain may change. In turn, this may lead to `As` invalidity and `Rs` modification or removal. For this, we use Poisson processes to model the events that constitute the probability of change in volatility of `A` and `R` at various time in the development process. In addition, we use stochastic processes to model changes in importance of the `Rs` during the development process. Finally, in order to model the validity of `A` and desirability of `R` at the start of the development process, we use a Bernoulli distribution.

Four, the invalidity risk is measured in terms of specific *metrics*: **Validity** $V(k, t)$, **Importance** $I(k, t)$, **Children weight** $C(k, t)$, and **Use-cases participation weight** $U(k, t)$, for k-th set member at time t (see [1] for details). The modeling is performed using a Monte Carlo approach – we simulate the dynamics of the BN from initial time until the final time specified by the user, by applying to it the processes and distributions described above. Risk metrics are applied to the BN at the final state to obtain the measure of project risk associated with invalidity of `A` and `R`.

Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)
0-7695-2425-7/05 $20.00 © 2005 IEEE

3 Simulation Example

Consider that an ATM Banking system needs access to the database of bank clients. The system must be operational one year from now. Let us denote this requirement as r_0. Two groups of stakeholders gave the following Rs: we have to implement the system using a centralized database (model X1), requirement r_1; or a distributed database (model X2), requirement r_2. Clearly r_1 and r_2 are conflicting Rs. The stakeholders gave the following As underlying r_1: a_1 – the developers are proficient in implementing X1, a_2 – X1 will handle the heavy transaction load; and the following As for r_2: a_3 – the developers are proficient at implementing X2; a_4 – X2 will handle the heavy transaction load. We also add a single assumption to r_0: a_5 – we assume that one year term given for implementation is a strict deadline. We may also assume that the invalidity of a_4 will imply invalidity of a_2. The failure of r_1 or r_2 will lead to the failure of r_0.

Which of the two methods for implementing a single use-case is less risky? Let us assume that there is no relation between these sets of As and Rs, and the rest of the system. Thus, we can treat the use-cases as separate systems.

Note that these use-cases are mutually exclusive. We model the dynamics of $I(k,t)$ using a diffusion process. The properties collected from the stakeholder are given in [1]. Both use-cases will have r_0 in them. $\dot{r}_0$ and $\ddot{r}_0$ denote this requirement under X1 and X2 respectively. Both instances of requirement r_0 will have the same properties at the start time. The requirements sets for the two cases are $\mathcal{R}_1 = \{\dot{r}_0, r_1\}$, and $\mathcal{R}_2 = \{\ddot{r}_0, r_1\}$. The relations between As and Rs are given in Figure 1.

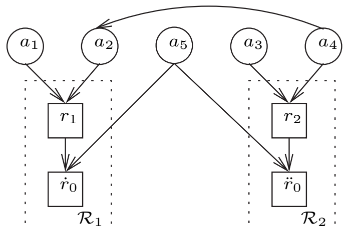

Figure 1. BN setup. Circles denote As, squares denote Rs. Arrows denote associative relationships (of different types [1]).

We simulate the system behaviour from $t_0 = 0$ until $T_f = 1$ with a weekly time step $\Delta t = 1/52$. As and Rs are elicited incorrectly with probability 0.02. We also assume that failure of any parent standard node will lead to an increase in child node failure intensity by 10%.

The average values of metrics for all requirements are obtained from ten thousand realizations of the system. We re-run each system realization simulation one hundred times to obtain standard deviation (sd) measurements. Cumulative measures for requirements in $\mathcal{R}_1$ and $\mathcal{R}_2$ are obtained by summing up the metric values for each of the requirements in the use-case. The smaller the value, the bigger the risk. The metric values are given in Table 1. Due to the lack of space we consider here only two metrics (validity and importance). The dynamics of the metrics over time is given in Figure 2. We see that at initial time $V_{\mathcal{R}}(\mathcal{R}_2, 0) > V_{\mathcal{R}}(\mathcal{R}_1, 0)$ and $I(\mathcal{R}_2, 0) > I(\mathcal{R}_1, 0)$. However, at final time, $V_{\mathcal{R}}(\mathcal{R}_2, 1) < V_{\mathcal{R}}(\mathcal{R}_1, 1)$ and $I_{\mathcal{R}}(\mathcal{R}_2, 1) > I_{\mathcal{R}}(\mathcal{R}_1, 1)$. This tells us that the invalidity risk associated with implementation of X2 would be higher than the one associated with X1. On the other hand, the importance $I(\mathcal{R}_2, 1)$ of requirements in $\mathcal{R}_2$ is still higher than in $\mathcal{R}_1$. Based on this, management can decide whether to implement $\mathcal{R}_1$, which has less invalidity risk, or to implement $\mathcal{R}_2$, which is deemed more important at time T_f.

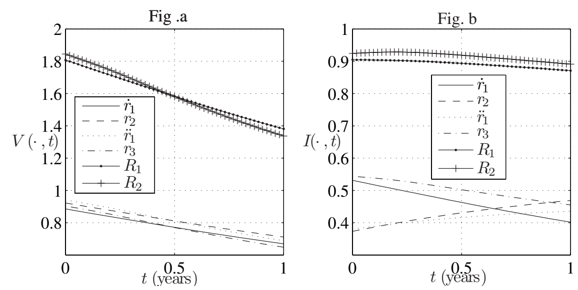

Figure 2. Fig. a: $V(\cdot, t)$; Fig. b: $I(\cdot, t)$.

Table 1. Metrics values ($\pm$ sd)

	$V(\cdot,0)$	$V(\cdot,1)$	$I(\cdot,0)$	$I(\cdot,1)$
$\mathcal{R}_1$	2	1.381±0.005	1	0.871±0.003
$\mathcal{R}_2$	2	1.337±0.004	1	0.891±0.003

In summary, we present a quantitative model of assumptions and requirements, and show how they change over time and what invalidity risk exists with these changes. We contend that this will aid in project management. The full poster presentation contains details from the paper supporting these ideas.

References

[1] A. Miranskyy, N. Madhavji, M. Davison, and M. Reesor. Modelling of Assumptions and Requirements Relations in the Context of Project Risk. Technical Report 645, Department of Computer Science, UWO, London, ON, Canada, 4 2005.

[2] A. van Lamsweerde and E. Letier. Handling Obstacles in Goal-Oriented Requirements Engineering. *IEEE Trans. Softw. Eng.*, 26(10):978–1005, 2000.

Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)
0-7695-2425-7/05 $20.00 © 2005 **IEEE**