Managing Requirements Invalidity Risk

Andriy Miranskyy[a] and Nazim H. Madhavji[b]
[a]Department of Applied Mathematics, [b]Department of Computer Science,
University of Western Ontario, London, ON, Canada
[a]amiransk@uwo.ca, [b]madhavji@csd.uwo.ca

Abstract

The importance of dealing with various types of risk in project management has long been recognised. In this position paper, we argue for monitoring and managing the risk associated with an application's domain changes which, in turn, may necessitate requirements modification or deletion during the development stage. In other words, we are concerned with how "valid" the requirements are through the development stages. We call this requirements invalidity risk. Based on the measure for calculating invalidity risk, we discuss how this risk can be managed in a typical risk management process.

1 Introduction and Position

It has long been known that the customer and end-user satisfaction with a software system[1] tends to diminish over time unless the system is upgraded (or evolved) on a regular basis [7, 10, 12] to keep up with the changing needs of the various stakeholders. The rate of satisfaction reduction would depend, in part, on the application type and the domain in which it is used. One can even say that, in general, the only software that does not change is *dead* software.

The phenomenon of stakeholder dissatisfaction with a software system is not just of theoretical interest. Consider the rather common situation today where an organisation is using software systems (which are infrastructural assets of sorts) to achieve its goals. These assets lose their net worth if they are not upgraded regularly. From a taxation point of view, this is quite well recognised to the extent that software depreciation is typically allowed at the rate of approximately 40% per annum (or has a life of approximately two and a half to three years) [1, 5].

This work is supported, in part, by NSERC, Canada.

[1]The ideas and concepts described in the paper apply, in general, to both pure software and hybrid hardware-software systems.

So, the fact that non-evolving software has a shrinking value over time (either in terms of stakeholder satisfaction or economics), begs a question as to by how much the "idea" of a software system, as denoted by its requirements, at the start of the project (both in the case of new systems as well as new releases), loses its worth by the time the system is released for use. While there is no single or clear answer to this question (because it depends on many factors, such as, the type of software, market competition, the extent of need by the customers, development cycle-time, and others), the high rate of depreciation *after* the release of the system is a good indicator that the software begins to lose its value even during its *development* phase.

Reasons for such degradation include changes in the domain of application, such as assumptions (underlying the requirements) which might have been valid at project start but may not all be valid at project end, and likewise for the requirements themselves. We call such degradation as *requirements invalidity*. Requirements invalidity is a *risk* because its extent is not known a priori.

From the arguments made, we take the position that it is important that requirements degradation and the ensuing invalidity risk, during a particular cycle of system development, is monitored and managed appropriately. Being aloof about this risk could lead to a rude awakening at the time of the system's release.

Of course, requirements invalidity risk is not the only risk to be managed in a software project. There are other, more well-known, direct and indirect project risks such as developer motivation with the system, customer relationships, technical glitches, delays, staff movement, unexpected budgetary constraints, and others (see, for example, [4]). Thus, requirements invalidity risk needs to be managed along with these other risks. In this paper, we describe some key issues to be considered when managing requirements invalidity risk in a typical risk management process.

2 Background

Below, we first describe some requirements risk factors, followed by our approach to determining invalidity risk.

2.1 Requirements Risk Factors

In [13], Wiegers mentions the following risk factors associated particularly with software requirements:

- lack of clear product vision
- lack of agreement on product requirements
- unprioritised requirements
- new market with uncertain needs
- new applications with uncertain requirements
- rapidly changing requirements
- ineffective requirements change management process
- inadequate impact analysis of requirements changes

In order to address requirements invalidity risk, we focus on two of the seven factors listed above: rapidly changing requirements (a product issue) and ineffective requirements change management process (a process issue). These are dealt with in our invalidity risk model [8] which we overview in the next section.

2.2 Invalidity risk model

There are four key aspects to our model:

1. Assumptions (A) and requirements (R) are explicitly represented and there exist interactions between assumptions and requirements. Three general types of interaction are A $\leftrightarrow$ A, R $\leftrightarrow$ R, A $\leftrightarrow$ R, as described in [8, 9][2].

2. Let $V_{(\cdot)}(j,t)$ be a binary variable, which is set to 1 when the j-th assumption (resp. requirement) in set $(\cdot)$ is valid (resp. desirable) at time t and is 0 otherwise. The interaction between assumptions and requirements is modeled (through the development cycle) using a Boolean Network (BN) framework [6]. This describes the connections between assumptions and requirements as well as the factors which cause them to change state.

3. The requirements 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 assumptions and requirements, respectively, at any given time. During system development (and usage as well, of course), the operational domain of the system may change. In turn, this may lead to assumptions invalidity and requirements 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 times in the development process. In addition, we use stochastic processes to model changes in the importance of requirements 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.

4. The requirements invalidity risk is measured at various points in the development project, as a n-tuple, in terms of specific *metrics*: **Validity** $V(k,t)$ (as a probabilistic value for the validity of a requirement or an assumption), **Importance** $I(k,t)$ (obtained from stakeholder input), **Children weight** $C(k,t)$ (based on dependency of a requirement on other requirements and assumptions), and **Use-cases participation weight** $U(k,t)$ (ubiquity of a requirement), for k-th set member at time t (see [9] for details). For a set of requirements we can obtain a single value by summing up the values for each of the metrics for all the requirements in the set. The management can then interpret the risk measures in the context of the project at hand.

The modeling of the invalidity risk is performed using the Monte Carlo approach — we simulate the dynamics of BN from the initial time until the final time specified by the user, by applying to it the processes and distribution described above. Risk metrics are applied to the BN at the final state to obtain the measure of invalidity risk associated with A and R. The reader is referred to [8, 9] for more detail on the core technical aspects of this work. Below, we now discuss how these measures can be incorporated into the risk management process.

3 Managing the Invalidity Risk

Risk management consists of a number of key generic steps: identification, analysis, planning, tracking and controlling; for examples see [13, 14]. An actual risk management process is not generally a linear process through these steps. Rather, it typically involves back-and-forth iterations among these steps. Below, we describe the focus in each of the steps of the process.

In the identification step, all risks associated with the project are identified. For invalidity risk, our model described above is operationalised and risks associated with individual requirements (or sets) are identified. Those requirements that are beyond a certain threshold for a given metric (determined by management decisions) are identified for further analysis.

In the analysis step, the identified risks are assessed for possibility of mitigation. For the risky requirements, the specific metrics described earlier are analysed. For example, for a requirement with high invalidity risk, the manage-

[2]Technical report [9] can be downloaded at `http://publish.uwo.ca/~amiransk/tr645.pdf`.

ment (possibly in conjunction with the appropriate stakeholders) needs to decide whether to modify or remove the requirement or leave it intact. In cases where system goals are severely affected by requirements removal, alternative (sets of) requirements may be considered that lead to the same system goals. In many cases, project (product) goals do change midstream; still, requirements at that time should be aligned with the new goals and invalidity risk should be assessed. Thought needs to be put on how to avoid the impending risks. Similarly, the fate of the requirements beyond the metric-specific thresholds needs to be decided upon by management. This is not an automatic process because requirements risk needs to be considered in the context of many development, business and customer related parameters. This is clearly a knowledge-intensive task and specific to the system being developed.

In the planning step, the mitigation actions are carried out based on the analysis from the previous stages. Tasks are prioritised and resources are allocated. From the point of view of invalidity risk, the goal is to identify the optimal set of requirements based on information analysis.

In the tracking step, we need to monitor the risk indicators. As soon as new information about a requirement or an assumption arrives, or on a periodic basis, the model is fed in with the new inputs and re-run to obtain the most recent invalidity risk measures.

Finally, in the controlling step, any deviations from the risk mitigation plan are corrected. Based on the tracking information, we may reconsider the decisions made at the planning stage.

In summary, in practice, invalidity risk is handled in a reactive way at best, often late in the development cycle. Proactive management of invalidity risk is (a) not a norm and (b) not formalised in any way. Requirements are, more often than not, frozen so that implementation can be carried out and system delivered. This clearly compromises the validity of the system at delivery time. This is why agile methods are quite appealing for they allow requirements changes deep into the development cycle. But not all projects are, or likely to be, carried out using agile methods. What we are proposing is a way to deal with requirements changes in a proactive way and a way for management to move forward knowing explicitly the risks involved based on simulation results. This is akin to installing fog-lamps on a motor car to deal with foggy driving conditions – something analogous we suffer in large and complex software projects. Let us now look at applicability of the approach.

4 Applicability of the approach

There are at least two important points to note in considering the applicability of our model. One is technical in terms of how the inputs to the model are obtained, the operational process for the model and where the output goes. The second is organisational, particularly in terms of management support.

4.1 Technical considerations

In order to run model simulation, each requirement and the underlying assumption needs to be represented in the model along with the associated properties such as probability of validity and importance through the development stages, etc. These parameters may be obtained from stakeholders or experts. The set of opinions obtained during a stakeholder survey can be small, which makes it difficult to determine the form of particular processes and the distribution to be used. In this case, Bayesian statistical techniques for elicitation of experts beliefs may be used; see [11, pp.158-161] for the list of references.

The data collection process seems to fit better the incremental approach, since batch input of parameters for all requirements and assumptions can be voluminous and tedious. Automatic input during elicitation would relieve this stress and eliminate transcription errors.

Addition of a new requirement or assumption to an existing set does not significantly increase the load on the user. However this insertion needs to be carried out carefully so that relations between old and new requirements and assumptions are correctly identified so as to obtain realistic output. Note here that by introducing the risk modeller, we need to pay particular attention to the correctness of the input information because management will likely believe the, otherwise, flawed output and base further projects decisions on this. It some cases, this can lead to costly or irreparable project damage.

In contrast to this, in traditional development, while not as technologically driven (which has its own, well-known, perils!), there is human-oriented flexibility to some degree to recover in the downstream development processes from inaccurate requirements and assumptions. Further work clearly needs to be carried out on the project-specific benefits and costs of adopting our approach in practice.

In any case, once the data is fed into the model, the simulation is performed automatically; the only parameter that user has to specify is the desired level of accuracy for calculations. The results obtained can be saved as a table of metric values for each requirement, which can be aggregated based on user preferences.

This table can be exported to third party project and risk management tools in order to consider them along with other risk factors (e.g., development cost and implementation time) for added benefit. For example, the tool Easy WinWin [3] (used for requirements negotiation) has a set of values such as ”Business Importance” and ”Ease of Realization” for each requirement. To incorporate invalidity risk, the Easy WinWin can possibly be extended to incorporate the invalidity n-tuple (metrics) as a column with four subcolumns. This would then give the stakeholders an added insight for requirements negotiation and into project management.

4.2 Organisational considerations

It would be naive to adopt the proposed modelling approach if there is no organisational awareness or support for the management of requirements invalidity risk. It is quite well-known from process assessments using the Software Capability Maturity Model [2] that approximately 5% of the organisations are classified at the "Initial" level; approximately 34% at the "Managed" level; approximately 27% at the "Defined" level; approximately 4% at the "Quantitatively Managed" level; and approximately 21% at the "Optimising" level.

The quantitative approach to software development and project management are a hallmark of the "Quantitatively Managed" and "Optimising" levels, suggesting that our approach would fit quite well with these advanced levels of maturity. That said, organisations at the lower levels of maturity (especially, "Managed" and "Defined") are known to gather metrics and improve processes in an attempt to hoist themselves up to the higher levels. Also, there is considerable focus on project management improvement at the lower levels of maturity. This would suggest that the concept and awareness of requirements invalidity risk and management issues can indeed be introduced as something to look forward to in the future for such organisations.

5 Conclusion

In this position paper, we take the argue that it is important that requirements degradation during a particular cycle of system development is monitored and that their invalidity risk during this cycle is managed appropriately. Being aloof about this risk could lead to a rude awakening at the time of the system's release. We describe our approach to modelling and simulating requirements invalidity risk and how such risk can be dealt with in a typical risk management process. We also describe the applicability of our approach in practice.

6 Acknowledgements

We thank the anonymous reviewers for their helpful comments, which have helped us to improve the paper.

References

[1] Taxation Laws Amendment (Software Depreciation) Bill. In *Bills Digest*, number 117, Canberra, Australia, 1998-1999. Parliament of Australia.

[2] Process Maturity Profile. CMMI® v1.1. SCAMPI^{SM} v1.1 Class A Appraisal Results. 2004 Year End Update. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, March 2005.

[3] B. Boehm, P. Grünbacher, and R. O. Briggs. Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, 18(3):46–55, May-Jun 2001.

[4] M. J. Carr, S. L. Konda, I. Monarch, U. F. Carol, and W. F. Clay. Taxonomy Based Risk Identification. Technical Report (CMU/SEI-93-TR-6, Software Engineering Institute, Carnegie Mellon, Pittsburgh, PA, 1993.

[5] IRS. *Instructions for Form 4562. Depreciation and Amortization*. Internal Revenue Service, Department of the Treasury, USA, 2004.

[6] S. Kauffman. *The Origins of Order. Self-Organization and Selection in Evolution*. Oxford University Press, Oxford, 1993.

[7] M. M. Lehman and J. F. Ramil. Rules and Tools for Software Evolution Planning and Management. *Ann. Softw. Eng.*, 11(1):15–44, 2001.

[8] A. Miranskyy, N. H. Madhavji, M. Davison, and M. Reesor. Modelling assumptions and Requirements in the Context of Project Risk . *To appear in RE'05: Proceedings of the 13th International Requirements Engineering Conference*, 2005.

[9] A. Miranskyy, N. H. 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.

[10] V. Nanda and N. H. Madhavji. The Impact of Environmental Evolution on Requirements Changes. In *Proceedings of the International Conference on Software Maintenance*, pages 452–461, Montreal, October 2002.

[11] A. O'Hagan and J. Forster. *Kendall's Advanced Theory of Statistics: Bayesian Inference*, volume 2B. Arnold, London, 2nd edition, 2004.

[12] D. L. Parnas. Software aging. In *ICSE '94: Proceedings of the 16th international conference on Software engineering*, pages 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[13] K. Wiegers. Know your enemy: software risk management. *Softw. Dev.*, 6(10):38–42, 1998.

[14] R. C. Williams, G. J. Pandelios, and S. G. Behrens. Software Risk Evaluation (SRE) Method Description (Version 2.0). Technical Report CMU/SEI-99-TR-029, Software Engineering Institute, Carnegie Mellon, Pittsburgh, PA, 1999.