On the Uncertainty of Technical Debt Measurements

Clemente Izurieta, Isaac Griffith, Derek Reimanis, Rachael Luhr
Software Engineering Laboratory
Department of Computer Science, Montana State University
clemente.izurieta@cs.montana.edu, {isaac.griffith, derek.reimanis, rachael.luhr}@msu.montana.edu

***Abstract*—Measurements are subject to random and systematic errors, yet almost no study in software engineering makes significant efforts in reporting these errors. Whilst established statistical techniques are well suited for the analysis of random error, such techniques are not valid in the presence of systematic errors. We propose a departure from de-facto methods of reporting results of technical debt measurements for more rigorous techniques drawn from established methods in the physical sciences. This line of inquiry focuses on technical debt calculations; however it can be generalized to quantitative software engineering studies. We pose research questions and seek answers to the identification of systematic errors in metric-based tools, as well as the reporting of such errors when subjected to propagation. Exploratory investigations reveal that the techniques suggested allow for the comparison of uncertainties that come from differing sources. We suggest the study of error propagation of technical debt is a worthwhile subject for further research and techniques seeded from the physical sciences present viable options that can be used in software engineering reporting.**

Keywords- software quality and maintenance; error analysis; technical debt

I. INTRODUCTION

This line of investigation proposes an alternative to the way we analyze and report values that are subject to uncertainty[1]. Typically, given multiple calculations of a metric (i.e., technical debt), we use well-known statistical methods to analyze and compare them, with the standard deviation as a good approximation for uncertainty. This is allowed provided the uncertainty values are *random*; however not all uncertainties are random. Technical debt calculations may suffer from *systematic* errors that occur as a result of poor calibration of the tools used to measure the debt. For example, a tool that performs static analysis for detecting code smells may consistently report violations where there are none. Regardless of the number of times we repeat an experiment or increase the sample sizes, if the errors[1] are systematic, then we cannot use the standard deviation as an approximation for uncertainty. Unfortunately, detection of such systematic errors is extremely difficult. In the physical sciences it is common for laboratories to designate an instrument as having systematic uncertainty expressed as a percentage. Software engineering laboratories produce tools that are typically developed as prototypes without all the quality assurance expected of commercial products, yet these tools are used in empirical investigations. Due diligence suggests that we should also allow for systematic uncertainty of the various dependent and independent variables that our tools measure. Thus, measurements will have a random component and a systematic component of uncertainty. Even in the unlikely scenario where all uncertainties are random, using propagation techniques for all variables participating in calculations (dependent and independent) is a complementary methodology to statistical methods. We initially focus on technical debt measurements, as a significant amount of systematic errors can potentially exist. A number of tools are also available that are complex enough in how they perform their calculations and could potentially benefit from this line of inquiry. Examples of these tools include CLIO [7] for checking modularity violations, RBML compliance checker [8] that automatically calculates a distance between a realization of a design pattern and the intended design, and CodeVizard [9] that looks for code smells (as introduced by Fowler [10]). Our goal is to generate early feedback and suggest possible benefits from this research. We pose the following questions:

Q1: How do we investigate systematic errors when computing technical debt parameters?
Q2: Are the suggested uncertainty propagation techniques enough?
Q3: What source code testing strategies can we use to identify and detect potential sources of systematic errors?

II. TECHNICAL DEBT

By definition, there is uncertainty in the measurement of technical debt. The technical debt metaphor describes a situation in which developers accept quality compromises in the current release to meet a deadline (e.g. delivering a release on time) [1]. Curtis, Sappidi and Szynkarski [2] state that "there is no exact measure of Technical Debt, since its calculation must be based only on the structural flaws that the organization intends to fix," and not all organizations fix, are aware of, nor quantify technical debt with similar tools, techniques, algorithms or precision. Additionally, small changes to parameters can admit large fluctuations of technical debt calculations; thus revealing the sensitivity of final estimates. Few (if any) studies in software engineering report on the analysis of error, as well as its propagation during scientific experimentation. The ability to keep uncertainty at a minimum is critical in any scientific field.

[1] The words error and uncertainty are synonymous and are used interchangeably throughout the manuscript.

978-1-4799-0604-8/13/$31.00 ©2013 IEEE

Technical debt studies reveal that no measurements made are free of uncertainty. This is evident in the calculations performed on technical debt as reported by CAST [3], Letouzey and Ilkiewicz [4] in the SQALE methodology, and by the Software Improvement Group (SIG) software quality assessment method [12] based on ISO/IEC 9126 [13]. Thus, in order to increase rigor in software engineering measurements, it is important to adopt a *modus operandi* that allows for representation of uncertainties and teaches how those uncertainties are propagated through calculations made in ratio scales. This is especially important when the source of errors is not random. We describe the early stages of experimenting with techniques used in the physical sciences [5] and applying them to software engineering measurements. Specifically, we focus on the measurement of technical debt. We expect that by using these techniques, scientists will be able to:

i. Compare two or more measured values (from potentially different sources) against one another,

ii. Propagate errors (random or systematic) through all allowed operators defined by a ratio scale, and

iii. Identify ways to provide meaningful error calculations in expressions that involve multiple variables– each variable subject to its own uncertainty.

III. METHODS

A. Definitions

We provide abridged definitions for the technical debt variables [6] that are under early investigation to quantify the uncertainty of their measurement.

1) Technical Debt Principal

Refers to the effort required to complete a task that is left undone. A task is a representation of a technical debt item that runs a risk of causing future problems if left undone.

2) Technical Debt Interest Amount

Refers to an estimate of the amount of extra work that will be needed to maintain the software if a technical debt item is not repaid. Interest incurs a continuing cost to its associated item.

3) Technical Debt Interest Probability

Refers to the probability that the technical debt, if not repaid, will make other work more expensive over a given period of time or a release. Probability is time sensitive. For example, a debt item may have a higher probability of being addressed in the next major revision of a product than the current version.

4) Violation

Refers to violations of agreed upon solutions to design or coding practices. CAST software [3] uses an ordinal scale to classify violations into high, medium, and low.

5) Uncertainty and Error

Random errors or uncertainty in measurement theory are abundant and refer to the delta that exists between the expected value of a measure and its actual measurement. Errors can overestimate or underestimate the expected value of a measurement. For example, in a technical debt context, the estimation of costs associated with technical debt items are subject to error.

B. Uncertainty of a Measurement

We use operations and notation proposed by Taylor [5] to state technical debt metrics as follows:

$$measured\ value\ of\ TD_{principal} = (TD_{principal})_{best} \pm \partial_{TD}$$

and represents an experimenter's best estimate of technical debt (TD) principal with a margin of error or uncertainty of ∂_{TD}. The estimate lies between $(TD_{principal})_{best} + \partial_{TD}$ and $(TD_{principal})_{best} - \partial_{TD}$. We use the uncertainty term ∂_{TD} as an aggregation of both random and systematic errors. The calculations presented herein can be applied to both.

IV. UNCERTAINTY CALCULATIONS

A. Comparing Measures

Since results that report on a single measure are uninteresting, scientists typically compare two or more measurements against each other to show relationships between values. Technical debt literature is relatively new, and unlike in other scientific fields, *accepted values* (i.e., values accepted by a scientific community) of technical debt do not exist (e.g., the ideal level of quality of a design pattern). Suppose two organizations A and B report their $TD_{principal}$ as follows:

$$(A\_TD_{principal})_{best} \pm \partial_{A\_TD}$$

$$(B\_TD_{principal})_{best} \pm \partial_{B\_TD}$$

then the estimate for the highest probable value of the difference is computed as follows:

$$(A\_TD_{principal})_{best} - (B\_TD_{principal})_{best} + (\partial_{A\_TD} + \partial_{B\_TD})$$

and the estimate for the lowest probable value of the difference is computed as follows:

$$(A\_TD_{principal})_{best} - (B\_TD_{principal})_{best} - (\partial_{A\_TD} + \partial_{B\_TD})$$

Assuming that both organizations report the uncertainty associated with their respective measurements using the same number of significant figures; then it is important that the discrepancy in uncertainty is also reported using the same number of significant figures. If one organization reports its uncertainty measurements with significantly more precision than another, then we must use the coarser uncertainty measurements to report results.

B. Propagation of Error

When computing the value of technical debt, we must also account for how the uncertainties of technical debt interest and probability (i.e., the independent variables) propagate. Even when we calculate the value of technical debt principal alone we must account for uncertainties in the average hours needed to fix violations (high, medium and low), the dollar cost per hour, and the average number of hours required to fix a violation. We use Taylor's [5]

rules to estimate the propagation of technical debt uncertainty. To exemplify the methodology we use the calculations proposed by Nugroho et al. [11].

1) Sums and Differences

The propagation of uncertainty in the sums or differences of measured quantities is similar to the example from section IV.A. Thus, if several quantities x_1 .. x_n with corresponding uncertainties $\partial x_1 .. \partial x_n$ are measured with uncertainty, then the overall uncertainty of their additions, differences, or combination of both operations is given by:

$$Uncertainty = \partial x_1 + .. + \partial x_n$$

2) Products and Quotients

The propagation of uncertainty in products or quotients of measured quantities is best given using *fractional uncertainty* notation. Recall from III.B the calculation of technical debt as:

$$measured\ value\ of\ TD_{principal} = (TD_{principal})_{best} \pm \partial_{TD}$$

then, the fractional uncertainty in the $TD_{principal}$ is defined as:

$$fractional\ uncertainty\ TD_{principal} = \frac{\partial_{TD}}{|(TD_{principal})_{best}|}$$

Nugroho et al. [11] calculate technical debt principal by estimating the Repair Effort (*RE*) necessary to increase the quality of the software to an ideal level. *RE* is then calculated as follows:

$$RE = RF \times RV$$

where *RF* is a Rework Fraction and *RV* is the Rebuild Value. Multiplying the System Size (*SS*) against a Technology Factor (*TF*) carries out the *RV* calculation for a system. *TF* is calculated as the number of man-months per source code statement, and *SS* can be calculated either by using lines of code (LOC) or function points. There is a large possibility of having uncertainty in all these variable calculations; for example, using function points carries a significantly higher degree of uncertainty than using LOC. Thus, if *RF* is given by:

$$measured\ value\ RF = RF_{best} \pm \partial_{RF}$$
and
$$measured\ value\ RV = RV_{best} \pm \partial_{RV}$$

then if the uncertainty for the measured value of *RE* is given by ∂_{RE}, it follows that:

$$\frac{\partial_{RE}}{|RE_{best}|} = \frac{\partial_{RF}}{|RF_{best}|} + \frac{\partial_{RV}}{|RV_{best}|}$$

In general, if several quantities x_1 .. x_n with corresponding uncertainties $\partial x_1 .. \partial x_n$ are measured, then the overall uncertainty of their products, quotients, or combination of both operations is given by:

$$\frac{Uncertainty}{|(measure)_{best}|} = \frac{\partial x_1}{|x_{1best}|} + .. + \frac{\partial x_n}{|x_{nbest}|}$$

3) Power Uncertainty

Nugroho et al. [11] calculate the technical debt interest amount as the difference between the *ideal* level of Maintenance Effort (*ME*) needed for a software module and the current level of maintenance effort that may include extra costs as a result of additional quality issues. The *ME* is given by the formula:

$$ME = \frac{MF \times RV}{QF}$$

where $QF = 2^{((qualityLevel-3)/2)}$, and quality levels range from 1 to 5, thus producing *QF* values of 0.5, 07, 1.0, 1.4, and 2.0 respectively. *MF* is a Maintenance Fraction representing the number of lines that undergo change in a year, and *RV* is the Rebuild Value (originally described in section IV.B.2); which can also be calculated as:

$$RV = SS \times (1 + r)^t \times TF$$

thus; for time *t* and growth rate *r*, the rebuild value (*RV*) of a system increases over time if left unattended. If the rate *r* changes as time increases, then this formula must account for the range of that movement, or the uncertainty of the measurement. To demonstrate how to account for this uncertainty we only focus on the $(1 + r)^t$ factor of the *RV* calculation. The multiplication by *SS*, and *TF*, subject to uncertainty is carried out using the propagation techniques for products and quotients described in section IV.B.2. If *r* is measured with uncertainty ∂r, then the uncertainty of the calculation of the factor $(1 + r)^t$ is given by $t \times \frac{\partial r}{r}$ for a fixed value of *t*. The overall fractional uncertainty of the *RV* calculation is then given as:

$$\frac{\partial_{RV}}{|RV_{best}|} = \frac{\partial_{SS}}{|SS_{best}|} + t \times \frac{\partial r}{r} + \frac{\partial_{TF}}{|TF_{best}|}$$

4) Technical Debt Interest Probability

Since the probability of interest varies with different time frames, a time element must be attached to the probability. Additionally, the value assigned to interest probability tends to be classified in an ordinal scale based on historical data. There are no definitive technical debt interest probability calculations; however if ratio scale arithmetic is used, then propagation of error can be accounted for with the proposed techniques. If however probabilities are table based and ordinal, further exploration is necessary.

C. Multivariate Uncertainty

Taylor [5] describes formulas that use quadrature where appropriate, but these techniques need validation in the technical debt domain. Quadrature allows us to discount the negligible effect of some unlikely error propagation possibilities, thus providing more realistic error ranges

when calculating the error of a multivariate expression. For example, in section IV.B.1 we described the calculation of $TD_{principal}$, where the expression is only made up of sums and differences. We showed that the total uncertainty equaled the sum of the uncertainties of each component in the expression. However; this formula clearly overestimates ∂_{TD} because this can only occur when we underestimate the values of each component in the expression by the full amount. Similarly we would underestimate ∂_{TD} when we overestimate the value of each constituent component in the expression by the full amount. As the number of components in the expression grows it is more unlikely that all participating components either overestimate or underestimate their values at exactly the same time. If the expression shown in section IV.B.1 only had two components (i.e., two variables) that represent the technical debt measurements of organizations A and B; given by $A\_TD_{principal}$ and $B\_TD_{principal}$, then there is only a 50% chance of underestimation or overestimation. The error is then given by $\sqrt{{\partial_{A\_TD}}^2 + {\partial_{B\_TD}}^2}$, which is always less than $\partial_{A\_TD} + \partial_{B\_TD}$. Quadrature is an applicable calculation when measurements come from Normal or Gaussian distributions and the measurements are independent. Can we then be justified in using quadrature in software engineering technical debt calculations that are subject to systematic errors? Or do we need other techniques to find better approximations of errors?

V. CONCLUSIONS

Unlike the physical sciences where we can, for example, use multiple clocks or meters to identify an artifact that produces systematic errors (e.g., a clock that is 5 seconds fast, or a meter whose readings are consistently low), technical debt tools can not be compared against other tools because they all compute equations differently, thus making detection of systematic errors extremely difficult. Until accepted values for certain calculations exist, or known reliable tools that we can calibrate against exist, we must seriously consider reporting results with their corresponding uncertainty, and we must also be aware of how these uncertainties propagate. This research seeks answers and potentially encouraging lines of inquiry to help answer the three questions posed in the introduction. To answer **Q1**, we argue that in the absence of established accepted technical debt values (principal, interest and probability), we can use reported uncertainties associated with these values as a means to investigate the existence of systematic errors by comparing the overlap of the uncertainties. This can help identify calibration issues when comparing outputs from multiple tools. In section IV.B we described the various ways in which the propagation of errors can occur. We suspect that additional propagation techniques are required in order to minimize errors and help answer **Q2**. For example, Taylor [5] describes formulas that use quadrature where appropriate, but these techniques need validation in the technical debt domain. Quadrature allows us to discount the negligible effect of some error propagation factors, thus providing more realistic error ranges. Answers to **Q3** remain elusive, however the propagation of errors from experimentation could be automated, and final estimates of debt could be compared against *expected values* that have organizational context and are stored in testing golden files. Seaman et al. [6] state that "in any approach to decision making about technical debt, some human intervention is required to provide information that cannot be reliably measured," thus understanding how the propagation of uncertainty occurs is a critical factor if we want to continue to improve the decision making process of which items to refactor.

REFERENCES

[1] W. Cunningham, "The WyCash Portfolio Management System," in *Addendum to the proceedings on Object-oriented programming systems, languages, and applications*, 1992, pp. 29-30.

[2] Curtis, B.; Sappidi, J.; Szynkarski, A., "Estimating the Principal of an Application's Technical Debt," *Software, IEEE* , vol.29, no.6, pp.34-42, Nov.-Dec. 2012, doi: 10.1109/MS.2012.156

[3] ——, *CAST Report on Application Software Health*, tech. report, CAST Software, 2011.

[4] Letouzey, Jean-Louis; Ilkiewicz, Michel, "Managing Technical Debt with the SQALE Method," *Software, IEEE*, vol.29, no.6, pp.44-51, Nov-Dec. 2012.

[5] Taylor, J. R. 1997. *An Introduction to Error Analysis*. University Science Books, 2nd. Edition, ISBN-13: 978-0935702750.

[6] Seaman, C.; Yuepu Guo; Izurieta, C.; Yuanfang Cai; Zazworka, N.; Shull, F.; Vetro, A., "Using technical debt data in decision making: Potential decision approaches," *Managing Technical Debt (MTD), 2012 Third International Workshop on* , vol., no., pp.45-48, 5-5 June 2012. doi: 10.1109/MTD.2012.6225999

[7] S. Wong, Y. Cai, M. Kim, and M. Dalton, "Detecting software modularity violations," in *Proc. 33rd International Conference on Software Engineering*, May 2011, pp. 411–420.

[8] S. Strasser, C. Frederickson, K. Fenger, C. Izurieta, "An Automated Software Tool for Validating Design Patterns," in *Proc. 24th International Conference on Computer Applications in Industry and Engineering*, November 2011, Honolulu, HI.

[9] N. Zazworka, "CodeVizard: a tool to aid the analysis of software evolution," in *Proc. Of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement,* September 2010, Bolzano-Bozen, Italy.

[10] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*, 1st ed. Addison-Wesley Professional, Jul. 1999.

[11] Nugroho, A.; Visser, J.; Kuipers, T., "An empirical model of technical debt and interest," In *Proceedings of the 2nd Workshop on Managing Technical Debt* (MTD '11). ACM, New York, NY, USA, 1-8. doi:10.1145/1985362.1985364

[12] Heitlager, I; Kuipers, T.; Visser, J., "A practical model for measuring maintainability." In *6th Internation Conference on the Quality of Information and Communications Technology*, 2007. QUATIC 2007. Pages 30-39.

[13] International Organization for Standardization. ISO/IEC 9126-1: Software Engineering – product quality – part 1: Quality model, 2001.