ISSN (e): **2250 – 3005** || *Volume, 07* || *Issue, 07*|| *July – 2017* ||
International Journal of Computational Engineering Research (IJCER)

Technical Debt Calculation and Its Uncertainties

[1]snigdha Mahiyashi Mohapatra, [2]sabyasachi Panigrahi
Gandhi Institute of Excellent Technocrats, Bhubaneswar, India
Oxford College of Engineering and Management, Bhubaneswar, Odisha, India

ABSTRACT
We don't know how much we owe (i.e., Technical Debt) unless a calculation is made. A lot of TD measurements have been proposed and they seem promising. There are mainly two different major approaches to calculate TD, one is estimation of TD with interest and the other is estima- tion without interest. However both of these approaches are not useful unless we understand and consider different un- certainties during calculation of these approaches. In this study, we will investigate (1) what kind of uncertainties are reported in literature related to Technical Debt and (2)why they are important to consider.

Categories and Subject Descriptors
D.2 [**Software**]: Software Engineering; D.2.9 [**Software Engineering**]: Management—productivity, programming teams, software configuration management

Keywords
Technical Debt, Principle, Interest, Interest Probability, Un- certainty, TD measurements

INTRODUCTION

The term Technical Debt (TD) was used for the first time by Ward Cunningham in 1992. It is a metaphor indicating technical compromises that produces short-term benefit but it hurts the long-term health of a software system [10]. Talk- ing in the context of source code technical debt is a poorly written code, requiring extra effort to correct and modify it in future. There are three terms related to technical debt including principal, interest amount and probability.

Principal: Principal of technical debt is defined as the cost of re-factoring to clean code.

Interest: Whereas, interest is defined as the extra costs andeffortrequiredbydeveloperstoworkwithmessycode /functionalityduringmaintenanceornewfeaturesaddition.

InterestProbability: InterestProbabilityisdefinedas the probability of technical debt leading to future problems andcosts,ifnotitisnotpaidontime.
Based on different issues and causes of technical debt Flower in [6] classified technical debt into different groups as presented in figure 1 above.

Figure 1: Types of Technical Debt, [6]

In literature and software industry different terms and properties are attributed to technical debt, Brown et al. dis- cussed some of the properties and major issues of technical debt [2] in detail to provide a vision to better understand technical debt which are are as follows:

Visibility: Due to lack of technical debt visibility, se- rious management and maintenance issues are caused. Sometimes maintenance task is handled by a third party organization, in this situation maintenance ven- dors are unaware of any unforeseen technical debt which was introduced by the development team. Moreover maintenance vendor do not get the idea how badly the code is written until maintenance is started. In case of identifying technical debt, it is hard for these mainte- nance vendors to analyze, that what were the causes of technical debt.

Value: Technical debt is not a negative thing, if it is managed wisely it can add value to software system. Just like, in real life mortgages besides its downsides help people to buy homes.

Present Value: Present value, cost and impact of the technical debt for the same project vary from time to time. Therefore it is necessary to analyze its im- pact probability and uncertainties during cost benefit analysis.

Debt Accretion: Managing technical debt is critical as too much debt leads to bad impact on the system. If technical debt is not paid on time, it become difficult to maintain and modify the system according to new requirement.

Environment: Technical debt is highly dependent to its developing environment and context. For exam- ple sometimes an organization deliberately introduces some technical debt to meet a short deadline or to over- come lack of cost and resources. Similarly, sometimes technical debt is incurred unintentionally due to lack of expertise of development team.

Origin of Debt: Technical debt can be classified into two broad categories based on its origin including strategic and unintentional debt. The former increases system value if it is managed wisely while the latter is highly discouraged as it badly impacts project value and management.

Impact of Debt: Both in case of strategic or acci- dental debt, the impact of debt is different varying in scope from local to global to overall software project.

Uncertainty: Estimating technical debt precisely is difficult, as it is dependent on different number of fac- tors, some of them has been discussed above. These factors vary from project to project. Additionally these factors also vary within a project over different time frames.

If TD is incurred wisely it helps to bring business value. However, this TD must be managed otherwise it causes se- rious issues in system maintenance and evolution [10]. In order to effectively manage TD, identification of TD items and TD measurement is necessary before prioritizing them in order to select which TD item should be paid first. For this purpose different TD estimation models have been pro- posed. However precision and accuracy of these measure- ment models varies and is effected by underlying uncertain- ties. These uncertainties are characterized due to lack of sufficient knowledge of different factors as discussed above, consequences and level of impact of these factors on system

health infuture.

As uncertainty in measurements is inevitable in software engineering processes [1].Its critical to consider different uncertainties and errors related to technical debt. The fo- cus of this research work is to provide an overview of these uncertainties. Paper organization is as follows: in section 2, two of the known models for technical debt are discussed followed by different uncertainty factors in detail and errors insection3andfinallyconcludingthepaperinsection4.

1. TECHNICAL DEBTCALCULATION MOD- ELS

In literature different methods have been proposed to cal- culate this technical debt. Following is the overview ofsome of thesemethods.

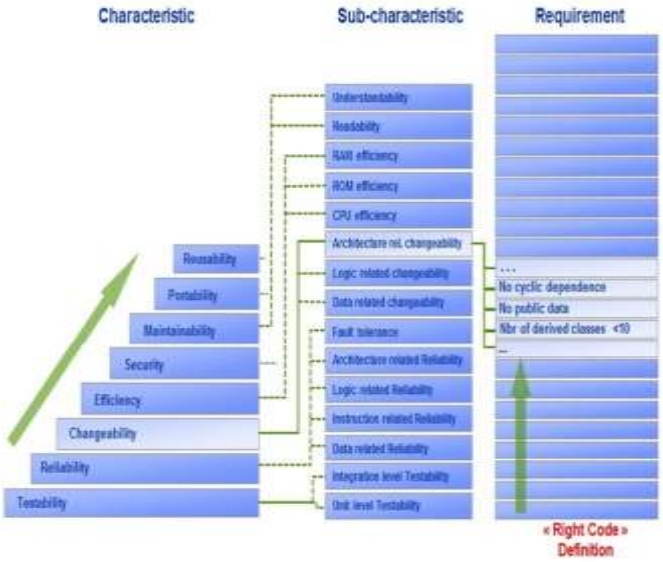

Figure 2: Chronological Order in SQALE, [9]

SQALE

Software Quality Assessment Based on Life cycle Expec- tations is a technique applied on source code for its quality evaluations. Version 1.0 of this method provide support for calculating technical debt in an organization. First of all, organization identifies the requirements related to projects, serving as the basis for right code. In SQALE it is called 'Quality model'. Any violation / non-compliance of this quality model creates the code debt. These requirements varyinnaturefromnon-functionalrequirementstocodepre- sentation, naming or design and architectural once. Against each requirement its remediation functions is also listed. These remediation functions provide the basis for calculat- ing remediation cost caused by non-compliance/violation of each requirement. Then, analysis tool is run to calculate the technical debt which is actually sum of all remediation cost incurredbytheviolationofabovesetrules.

TheSQALEmethodprovidestherequirementtobegrouped under eight quality characteristics including testability, re- liability, changeability, efficiency, security, maintainability, portabilityandusabilityinachronologicalorder.Thischron- icallyisimportantandshouldnotbechangedasitbadlyef- fectsandincreasethetechnicaldebtifitischanged.Figure 2provideoverviewchronologicallyorderedeightcharacteris- tics using by SQALE. If a requirement is related to multiple characteristics it must be associated with the lowest one in chronologicalorder.

AgainsteachoftheseeightcharacteristicSQALEprovides an indicator index to built a pyramid view and correspond- ing debt related to each characteristic debt. If the chronol- ogy order in the pyramid is not followed it leads to loss of resource and time, thus introducing overhead costs. For ex- ample testability should be done before targeting reliability. Thispyramidprovidethetechnicalperspectiveofthedebt.

As paying whole technical debt is not feasible most of the times. In this case, one need to prioritize within techni- cal debt available time. For this solution SQALE provides another method and different indicators and indexes. Just like remediation function non remediation function is also associated with each requirement. This the non remedia- tion function actualy estimates the penalty that the Product Owner (or someone who represents the Business) may claim ascompensationforacceptingviolations[9].Foreaseofuse eachrequirementisclassifiedintodifferentclassesofconse- quence such as ”blocking”, ”critical”, ”major” ”high” ”low” and a symbolic cost is assigned to each category. As it is difficult to find out exact consequence of each violation so webroadlyclassifyandassociateasymboliccostwithit.

SQALE then defines index for summing up all of the non remediation costs which is called SBII (SQALE Business Impact Index). This SBII provides the business perspective related to technical debt. So instead of just relying on the technical aspect SQALE takes into account business per- spective to better utilize and payoff it within limited time and cost constraint. Remediation priority (Non-remediation cost / remediation cost) is displayed in the form of Debt Map Graph which priorities giving with high return can be selected.

CAST: Curtis EstimateModels

Curtis et al. [3] presented three different estimate mod- els for calculating technical principle debt. This article was focused on calculating the technical debt as principal only. With the availability of limited resource and time, it is not possible to reduce all of the technical debt. Therefore com- panies need to prioritize to reduce technical debt based on their available budget. Authors provided the overview for calculating principle technical debt based on followingthree parameters:

1. No. of should fixviolations
2. Hours required to fix allviolations
3. Cost oflabor

Where hours required to fix all violations can be obtained from historical data of similar projects while cost of la- bor can be set as the average labor cost in an organiza- tion. These parameters can be used under different situa- tions varying from organization to organization such in ac- cordancewiththeseverityofviolationsuchashigh,medium and low severity. Based on these three parameters authors defined three estimate methods to calculate the technical debt as shown in Figure3.

For estimate method 1, constant hours were set to fix dif- ferent percentages of violations, authors marked it as too conservative approach. For estimate model 2, variant per- centage of violations were set to be completed in different time-span. While estimate model 3, considered different hours distribution to fix high level violations. To find the effectiveness of each estimated model, authors tested them using data from Appmarq benchmark repository maintained by CAST Software for 700 different applications containing at least 10 KLOC per application. These applications were

Variable	Parameter values		
	Estimate 1	Estimate 2	Estimate 3
Violations that must be fixed			
High-severity violations	50%	100%	100%
Medium-severity violations	25%	50%	
Low-severity violations	10%		
Hours to fix			
High-severity violations	1 hour	2.5 hours	10%—1 hour 20%—2 hours 40%—4 hours 15%—6 hours 10%—8 hours 5%—16 hours
Medium-severity violations	1 hour	1 hour	
Low-severity violations	1 hour		
US$ per hour			
All violations	75	75	75

Figure 3: Parameter Values For Three Estimates, [9]

analyzed using CAST's Application Intelligence Platform (AIP). AIP is supported with databases of 1200+ violation rules for 28 different languages belonging to architectural and coding.

AIP analyzes and parses source code based on meta-data parsing. Violation score is defined by the probability for a rule being triggered and number of times it is violated based on severity. Then different reports are generated to guide the developers to location where each volitional is done. AIP combinesviolationscoresinunderfollowingfivecategories, robustness, performance efficiency, security, transferability and changeability. Authors presented data for above dis- cussed model by diving and categorizing on the basis of lan- guage and then in the perspective of quality measure. Based on the analysis authors presented benchmark stats based on the historical information of 700 applications to define dif- ferent violation rules specific to eachcategory.

3. UNCERTAINTY IN TECHNICAL DEBT MODELS

There are a large number of TD measurementapproaches, models and tools available. Li et al grouped and classified 49differentsuchstudiesonTDmeasurementalongwithen- listing 8 different tools from research literature [10]. Some of these tools are mentioned as follows: CLIO tool [14] is used for for detecting modular violations, RBML checker is used [12] for calculating deviation between design pattern and actual implementation of that pattern. For detecting code smells, as defined by famous author Fowler[5] there is a tool called CodeVizard[15].

Ifwecalculatetechnicaldebtmultipletimeswithdiffer-enttools,itisobvioustogetdifferentresults.Because,each tools have different parameters, dependent / independent variables and algorithms to calculate technical debt.As

each tool operates on different TD calculation model and theremustbesomeuncertaintiesassociatedtoeachunderlyingmodel.Curtis,SappidiandSzynkarski[3]statedthat ganizations have calculated and reported TD. Lets say two companies A and B have reported their Technical debt prin- cipal as follows

"there is no exact measure of Technical Debt, since its cal- culation must be based only on the structural flaws that the organization intends to fix," as different organizations have

(A TD
principal
)best±∂

ATD
(2)

differentgoalssotheymeasureandallocateresourcestofind and fix technical debt differently to withscopes.

With the availability of such large number of TD measurementapproachesandtools,anorganizationoftenneed tomakedecisionforchoosinganeffectiveapproach/tool. Thus,selectinganappropriateapproach/toolwithoutknowingitsprecisionandrelateduncertaintyisadauntingchoice.

UncertaintiesareinevitableinSEprocessesandmeasurementmodels[1].Abranetalclassifieduncertaintiesofmea- surement into Experimental standard deviation, Error (of measurement)Deviation,Relativeerror,Randomerror,Systematicerror,Correction,andCorrectionfactor[1].While reporting measurement method / models, sufficientinforma- tion must also be provided regarding itsuncertainties.

Reports on technical debt mention such uncertainties for example,LetouzeyandIlkiewicz[9]intheSQALEmethodology mentionedit.TheSIG(SoftwareImprovementGroup) software quality assessment method [7] based on ISO/IEC 9126, also tells that technical debt measurements are not free of uncertainties. So in technical debt measurements it is very important to take into account of these uncertainties and accommodate them in expressing technicaldebt.

In order to accommodate these uncertainties in techni- cal debt calculation its vital to understand their causes and origins. However, in literature related to TD not much in- formation while discussing different TD measurements ap- proaches and models. Izurieta et al adapted different general uncertainties principles from physics and mapped them to TD models [8]. We will make an attempt to discuss these providedmodelsandelaboratetheminmoredetails.

measuredvalueofTDprincipal =(TDprincipal)best ∂TD
(1)

equation(1)definesthegeneralmatrixforTDmeasurement. This matrix also in addition to considering technical debt principal also take into account margin (denoted by ∂TD)of error or uncertainties of technical debt calculations. In above equation $TD_{principal})_{best}$indicates the best results reported by subjected tool / model. The uncertaintyterm ∂TD in above example catering both random andsystematic

$(B\ TD_{principal})best \pm \partial_B TD$ (3)

then for computing highest probable value for the estimate we can use this equation.

$(A\ TDprincipal)best(B\ TDprincipal)best+(\partial A\ TD + \partial B\ TD)$ (4)

and for computing lowest probable value for the estimates equation is

$(A\ TDprincipal)best(B\ TDprincipal)best(\partial A\ TD + \partial B\ TD)$ (5)

If we say both companies have described their uncertainty with measurements using equal number of figures then in- consistencyinuncertaintyshouldalsobeenlistedusingsame number of figures. If one company is reporting uncertainty with different granularity than the other then the final re- ports must take that into account and use common measure- ments inresults.

Propagation ofErrors

Whencalculatingcompletetechnicaldebtcalculations,we must take care of how value of uncertainties of technical debt interest and probability propagate. If we calculate the value of technical debt principal, we still need to take care of uncertainties present in average labor hours to fix low quality code which have architecture violations and other issues, the cost of per hour, and average cost in time to fix one violation. Again by using Taylor's [13] rules to estimate the propagation of technical debt uncertainty [8] discussed following equations proposed by Nugroho et al.[11].

Sum AndDifferences

If several quantities x1 ..xn with their uncertainties are measured with uncertainty then the overall uncertainty of their additions, differences or combinations of both opera- tions are:

$uncertainity=\partial x1+..+\partial xn$ (6)

Product AndQuotients

The propagation of uncertainty in measured quantities in context of products or quotients can be calculated by us- ing fractional uncertainty notation. Calculation of technical debt as defined in terms of equation 1 then we can define fractional uncertainty in technical debt principal asfollows:

errors.

3.1 ComparingMeasures

fractionalUncertainityTD

$$principal = \frac{\partial TD}{TD_{principal}best}$$ (7)

Results containing single measure are not that useful, sci- entists usually compare two or more measurements to check relationships between values. Technical debt literature is fairly new and the disadvantage is there is no standard ac- ceptedvaluesfor technical debt calculation like other sci- entific fields. Consider a scenario where an organization C purchasessoftwarefromorganizationAandassignorganiza- tion B for its maintenance immediately. Organization C has requested to both the organization to report TD estimates whenthepurchaseandmaintenancemade.Nowbothor-

RE = RFXRV

here RF is Rework Fraction and RV is Rebuild Value. The RV can be calculated for a system by multiplying System Size (SS) again Technology Factor (TF). Number of man- months per statement is TF and System Size (SS) can be calculated either by using lines of code (LOC) or function

points. Also uncertainty will be there in above calculations too like function point calculation will have higher degree of uncertainty than using LOC. So Rework Fraction can be calculated likethis

measuredvalueRF=$RF_{best}\pm\partial_{RF}$ (8)

measuredvalueRV=$RV_{best}\pm\partial_{RV}$ (9)

then uncertainty for the measure value of Repair Effort (RE) is given by

Formulas that use quadrature where appropriate are de- scribed by Taylor [13] but these need validation in technical debt domain. For ignoring negligible effect of some unlikely error propagation possibilities we can use quadrature, which will help us in having realistic error range when calculation error in multivariate expression. When measurements come from Normal or Gaussian distributions we can use quadra- ture equations nicely but those distributions also should be independent.

3.4 Technical Debt InterestUncertainty

$$\frac{\partial RE}{|REbest|} = \frac{\partial RF}{|RFbest|} + \frac{\partial RV}{|RVbest|} \quad (10)$$

Carlous et al. in [4] presented a cost analysis model for estimating technical debt using binary trees. Carlous etal. consideredtwoimportantparameterswhichareinterestun-

ifseveralquantitiesx1âĂęxnwiththeircorresponding

uncertainties then total uncertainty of their products,quo-

tients or both can be calculated like following

certainty and time frames. Interest uncertainty is defined as the probability that no extra cost is derived from technical debt [4]. Technical debt vary from time to time for the same

Uncertainity

$$\frac{\partial measure_{best}}{|measure_{best}|} = \frac{\partial x1}{|x1best|} \_\_\_\_\_ + \frac{\partial xn}{|xnbest|} \_\_\_\_\_ \quad (11)$$

projectundermaintenancedependingondifferentnumber

of factors. These factors differ in nature from internal such as low code complexity to external such as difference in use

PowerUncertainty

Nugrohoetal.[11]presentedtechnicaldebtinterestamount calculation as the difference between ideal level of Mainte- nance Effort (ME) needed for a software module and current level of maintenance effort. For calculating Maintenance Ef- fort formulais

of software for varying time length depending upon business activities.

Using maintenance cost as the measuring unit for tech- nical debt during system maintenance and evolution, if a software is not modified over a time period than no interest amount should be paid for this period. Taking in account

$$ME = \frac{MF X RV}{QF} \quad (12)$$

interest uncertainty factor helps to better estimate cost and benefits of technical debts. Here $QF = 2^{((qualityLevel-3)/2)}$ where quality level can have values from 1 to 5, so QF values can 0.5, 0.7, 1.0, 1.4 and 2.0. Maintenance Fraction (MF) is the number of lines that will be subjected to change in a year and Rebuild Value (RV) can be calculated as

$$RV = SSx(1+r)^{t} XTF \quad (13)$$

so in above equation for time and growth rater, the RV of a system will increase over time it âĂŹs not taken care of systematically. If the rater changes as time increases then this equation should take that into account too o uncertainty of the measurement. How it can account for uncertainty we can only focus on $(1+r)^{t}$ factor of the Rebuild Value (RV) calculation. The uncertainty in multiplication of System Size (SS) and Technology Factor (TF) is carried out using propagation techniques for production and quotients as described before. If r is measured considering uncertainty then the overall uncertainty of Rebuild Value (RV) can be calculated as follows

CONCLUSION

Technical debt is a popular term used to define technical compromises undertaken during the software development. If managed properly technical debt minimization can bring great value to the product. In order to manage and or- ganize technical debt different calculation models and ap- proaches are used. However precision of these approaches vary greatly due to large number of uncertainties. In order to get the most precise calculations these uncertainties must be taken into account during TD calculations. In this paper we have highlighted the importance and different classes of uncertainties that being used in the literature during TD calculations.

$$\frac{\partial RV}{|RV_{best}|} = \frac{\partial SS}{|RV_{best}|} + t X \frac{\partial r}{r} \frac{\partial TF}{|TF_{best}|} \quad (14)$$

Ninth International, pages 2–11. IEEE, 2003.

[2] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord,

Technical Debt Interest Probability

Because probability of interest will vary with respect to time so a time element is necessary to consider in calcula- tions more over values assigned to interest probability usually classified in ordinal scale based historical values. There is no systematic formula to calculate interest probability calculations, if probabilities are table based and ordinal then further exploration is needed to come up with a formula for calculations.

Multivariate Uncertainty

REFERENCES

[1]. Abran, A. Sellami, and W. Suryn. Metrology, measurement and metrics in software engineering. In Software Metrics Symposium, 2003. Proceedings.
[2]. Ozkaya, et al. Managing technical debt in software-reliant systems. In Proceedings of the
[3]. FSE/SDP workshop on Future of software engineering research, pages 47–52. ACM, 2010.
[4]. B. Curtis, J. Sappidi, and A. Szynkarski. Estimating the principal of an application's technical debt. IEEE software, (6):34–42, 2012.
[5]. C. Fernández Sánchez, J. Díaz Fernández,
[6]. J. Garbajosa Sopeña, and J. Pérez Benedí. A cost-benefit analysis model for technical debt
[7]. management considering uncertainty and time. 2013.

[8]. M. Fowler. Refactoring: improving the design of existing code. Pearson Education India, 1999.
[9]. M. Fowler. Technical debt quadrant. Bliki [Blog]. Availablefrom:http://www.martinfowler. com/bliki/TechnicalDebtQuadrant. html,2009.
[10]. Heitlager, T. Kuipers, and J. Visser. A practical model for measuring maintainability. In Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the, pages 30–39. IEEE,2007.
[11]. C. Izurieta, I. Griffith, D. Reimanis, and R. Luhr. On the uncertainty of technical debt measurements. In Information Science and Applications (ICISA), 2013 International Conference on, pages 1–4. IEEE,2013.
[12]. J.-L. Letouzey and M. Ilkiewicz. Managing technical debt with the sqale method. IEEE software, (6):44–51, 2012.
[13]. Z. Li, P. Avgeriou, and P. Liang. A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101:193–220,2015.
[14]. Nugroho, J. Visser, and T. Kuipers. An empirical model of technical debt and interest. In Proceedings of the 2nd Workshop on Managing Technical Debt, pages 1–8. ACM,2011.
[15]. S. Strasser, C. Frederickson, K. Fenger,and
[16]. C. Izurieta. An automated software tool for validating design patterns. In ISCA 24th International Conference on Computer Applications in Industry and Engineering. CAINE, volume 11,2011.
[17]. J. R. Taylor. An introduction to error analysis: The study of uncertainties in physical measurements, 327 pp. Univ. Sci. Books, Mill Valley, Calif,1982.
[18]. S. Wong, Y. Cai, M. Kim, and M. Dalton. Detecting software modularity violations. In Proceedings of the 33rd International Conference on Software Engineering, pages 411–420. ACM,2011.
[19]. N. Zazworkaand C. Ackermann. Codevizard: a tool to aid the analysis of software evolution. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, page 63. ACM,2010.