iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** XI **Month of publication:** November 2017

DOI: http://doi.org/10.22214/ijraset.2017.11100

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue X1, November 2017- Available at www.ijraset.com

Technical Debts, Impact and Settlement

Saurabh Dhupkar

Abstract: ***This paper focuses on studying 'Technical Debt' from the perspective of trade-offs between various system components and Software Quality Attributes, assuming that taking a technical debt is an involuntary action by stakeholders under unavoidable circumstances. For example, instead of considering lack of knowledge as a human factor for Technical Debt, this paper considers that lack of schedule and resources to gain the knowledge is the actual factor. This paper also considers the impact of Technical Debt on Life Expectancy of the software and its various factors.***
Keywords: Technical Debt, Software Quality, Software Quality Attributes, Software Life Expectancy

I. INTRODUCTION

Technical Debt is the term used to refer the scenarios, factors which contributed to it and the impact when any stakeholder chooses a quick fix over the right solution for any use case or modification in new or existing software. There are various factors and root causes of Technical Debt, similarly Technical Debt affects the software in various ways.

Technical Debt can be simplified as a compromise in one or more Software Quality Attributes for any reason. When any developer or architect chooses a dirty quick fix over the right solution, he/she is taking a debt from one or more Software Quality Attributes, which if not settled properly, can affect the software in various ways.

For example, to meet a scheduled due date a developer chooses to hard code a part instead of the right solution, he/she is compromising in various Software Quality Attributes like Reliability, Modifiability, Performance etc. Longer this quick fix stays in code, larger will be the impact. The impact can be seen in terms of factors like 'Cost per change', 'Efforts per change', 'Unexpected behaviour of software' etc.

Technical Debt is not really a defect or a bug in most of the cases, but a less that optimum solution. In some cases, the solution may even look best possible with the initial set of requirements, but with the next round of requirements come, the original architecture may start looking like inadequate one. Even if the architect or developers try to forecast as much as they can, there will always be a point where their forecast falls behind the requirements.

To survive, every software must maintain all its Critical Software Quality Attributes in the 'Acceptable Range'. Every software has its own acceptable range and its own Critical Software Quality Attributes depending on the domain, type of software etc. Critical Software Quality Attributes are basically of two types -.

A. Common Critical Software Quality Attributes

These are the very fundamental Software Quality Attributes that every software must maintain irrespective of the type and domain or other factors.

1) Reliability
2) Usability
3) Performance
4) Availability

B. Special Critical Software Quality Attributes

These are other Software Quality Attributes that are vital for the survival of the software. For example, Interoperability can be a Critical Software Quality Attribute for a software which is intended to work with multiple heterogeneous systems in terms of technology, domain and type etc.

Whenever a developer or architect chooses to take a Technical Debt from any of the Software Quality Attributes, the impact depends on the criticality of the attribute and the age of the debt. Whenever the developer chooses to prioritize a lower critical Software Quality Attribute above a higher Critical Software Quality Attribute he/she is taking a Technical Debt from the corresponding higher Critical Software Quality Attribute termed as 'Principal Quality Attribute' of the Technical Debt. However, in none of the cases of Technical Debt, only one Software Quality Attribute is involved. There is always some collateral damage and the other Software Quality Attributes participating in the Technical Debt are termed as 'Associate Quality Attributes' of the Technical Debt. The combination of Principal and Associate Quality Attributes in a Technical Debt completely depends on the

©IJRASET (UGC Approved Journal): All Rights are Reserved

scenario and the impact of the Technical Debt depends on criticality of the Software Quality Attributes involved along with the age of the Technical Debt.

C. Technical Debt Settlement Estimate

Technical Debt Settlement Estimate is time and/or cost required to settle the Technical Debt completely without introducing any more Technical Debts in the process.

1) Base Technical Debt Settlement Estimate: Base Technical Debt Settlement Estimate is the Technical Debt Settlement Estimate at the age 0 of the Technical Debt. This value basically represents the impact and spread of the Technical Debt. Higher the Base Estimate, worse is the Technical Debt to settle.

II. TYPES OF TECHNICAL DEBT

Keeping the assumption in mind that "No stakeholder willingly takes a Technical Debt", Technical Debts can be categorized by the trade-offs in which they are taken.

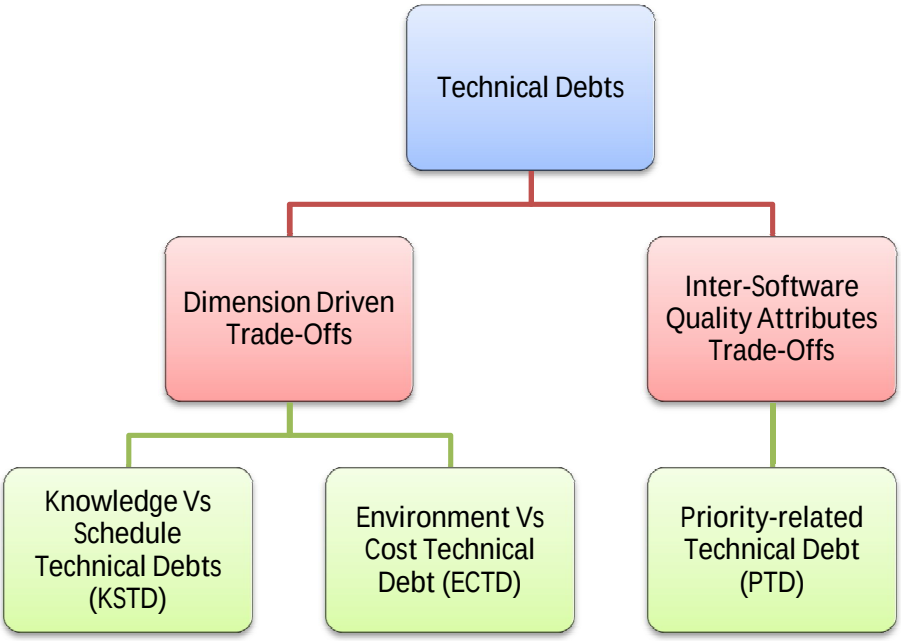


Figure 1 : Types of Technical Debt

The 'Dimensions' considered here are ‘Time’ and ‘Cost’ which are primary resources for business.

A. Dimension Driven Trade-offs

These trade-offs are the one where Time or Cost is involved, which are the fundamental resources of business. Therefore, this type of trade-offs can also be termed as ‘Software vs Business Trade-offs’.

With the increasing importance of software in business there is a corresponding increase in scenarios of Software vs Business Trade-offs. In most of the cases, where there is no show stopper issue involved, Business takes precedence over Software. In this kind of scenarios following types of Technical Debts are taken –

1) Knowledge vs Schedule Technical Debt (KSTD): In this kind of trade-offs, the stakeholders are not aware of the right solution and they are not in position of investing time in discovering it. Therefore, lack of knowledge is not the factor here instead it is lack of time to gain the required knowledge up to adequate level. Therefore, stakeholders choose to go ahead with the available knowledge and come up with the best possible solution within their knowledge range. Even though the stakeholders are aware that there can be a better solution, they are lacking the estimate in terms of time and cost to gain the required knowledge..

In this type of Technical Debt, ‘Time’ is the dimension that plays role. Time required to settle the Technical Debt is exponentially proportional with the age of the Technical Debt.

$$T \propto e^{a}$$

©IJRASET (UGC Approved Journal): All Rights are Reserved

Where

T : time required to settle the Technical Debt

a : age of the Technical Debt

$$T_{na} \propto e^{na}$$

Let's assume we are trying to estimate the cost required to settle the Technical Debt at the age n times more than the current.

$$T_{na} \propto e^{n} \times e^{a}$$

However, e^{a} is the Technical Debt Settlement Estimate at the age a

$$T_{na} \propto T_{a} \times e^{n}$$

Considering, the Base Technical Debt settlement estimation was done.

$$T_{a} \propto T_{0} \times e^{a}$$

That indicates thattime required to settle Technical Debt increases exponentially with the age of the Technical Debt multiplied by the Base Technical Debt Settlement Estimate.

2) Environment vs Cost Technical Debt (ECTD): In this kind of trade-offs, the stakeholders are mostly aware of the right solution but they are also aware of the limitations of their current environments. The most common examples are of the type where modern requirements are placed against very old software systems. Stakeholders are not in position to upgrade the environment to the adequate level due to cost factors.

In this type of Technical Debt, 'Cost' is the dimension that plays role. Cost required to settle the Technical Debt is exponentially proportional with the age of the Technical Debt.

$$C \propto e^{a}$$

Where

C : Cost required to settle the Technical Debt

a : age of the Technical Debt

$$C_{na} \propto e^{na}$$

Let's assume we are trying to estimate the cost required to settle the Technical Debt at the age n times more than the current.

$$C_{na} \propto e^{n} \times e^{a}$$

However, $e^{a}$is the Technical Debt Settlement Estimate at the age a

$$C_{na} \propto C_{a} \times e^{n}$$

Considering, the Base Technical Debt settlement estimation was done.

$$C_{a} \propto C_{0} \times e^{a}$$

That indicates that cost required to settle Technical Debt increases exponentially with the age of the Technical Debt multiplied by the Base Technical Debt Settlement Estimate.

Overall Impact Calculation: In some of the cases, the system is overloaded with previous Technical Debts which make it impossible for the stakeholders to implement the right solution and force them to settle by taking another debt.

In this kind of cases, a major clean-up project can improve the scenario. However, time and cost play a major role there too, redirecting stakeholders from clean-up to renovation which can lead to higher cost and time estimate.

$$T \propto \int T_{a} \propto \int T_{0} \times e^{a}$$

Equation 1 : Overall time required to settle all the Technical Debts

$$C \propto \int C_{a} \propto \int C_{0} \times e^{a}$$

Equation 2 : Overall cost required to settle all the Technical Debts

B. Inter-Software Quality Attributes Trade-offs

In this kind of scenario, the stakeholders have to trade-off between two or more Software Quality Attributes.

1) Priority-related Technical Debt (PTD): In this kind of Technical Debts, the stakeholders prioritize a lower criticality Software Quality Attribute above a higher criticality Software Quality Attribute. For example, where stakeholders are giving higher importance to 'Configurability' than 'Performance', where Performance is a Common Critical Software Quality Attribute while Configurability can be listed in 'Optional Software Quality Attributes'. In this type of example, the stakeholders are taking Technical

©IJRASET (UGC Approved Journal): All Rights are Reserved

Debt from 'Performance', which makes it 'Principle Quality Attribute' while there can be multiple 'Associate Quality Attributes' such as Readability, Modifiability etc.

III.IMPACT OF TECHNICAL DEBTS

A. Deciding Factors

Technical Debts affect the software in various ways. The deciding factors are -

1) Age of Technical Debt

a) Higher the age of the Technical Debt, higher is the impact

2) Combination of Principal and associate quality attributes

a) Impact depends on involvement of Critical Software Quality Attributes.

b) More critical attributes involved, higher is the impact..

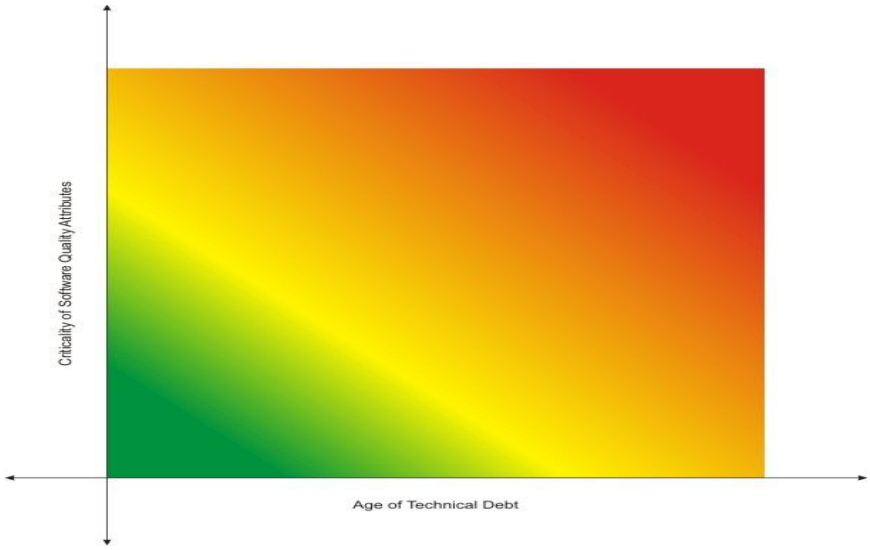


Figure 2 : Impact of Technical Debt with age and criticality of Software Quality Attributes

c) Coupling

d) Higher the coupling, higher the impact

Considering the module shown in red colour undergoes major changes, the impact propagates in other modules based on level of coupling. In the system with loose coupled modules the impacts on other modules decreases and impact propagation can be minimized.

B. Cost per change

This is the most impacted, quickly and easily visible and most importantly the easily quantifiable factor. It is a common belief that cost per change increases with the age of software. However, the Technical Debts accelerate the rate of increases in cost per change. The number and age of Technical Debts along with their respective combination of principal and associate quality attributes play vital role in increase in cost per change.

C. Decrease in Life Expectancy

Software Life Expectancy depends on value generated by the software vs the cost of maintenance. With the age, maintenance cost of software grows exponentially. However, with every Technical Debt the rate of increase in maintenance cost increases. The increment factor depends on number of Technical Debts and their age along with the Principal and Associate Software Quality Attributes involved.

To survive, every software must maintain all its Critical Software Quality Attributes in the acceptable range. However, every time a Technical Debt is taken from a Software Quality Attribute it starts losing its value and the rate of loss increases exponentially with the age of debt.

Once value of even one Critical Software Quality Attribute falls out of acceptable range, the software loses its 'Value Generation' and finally leads to decommissioning.

©IJRASET (UGC Approved Journal): All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue X1, November 2017- Available at www.ijraset.com

IV. TECHNICAL DEBT SETTLEMENT PROCESS

A. Proactive Technical Debt Settlement Process

1) Code review: The very basic difference between a defect and a Technical Debt is that the software can still provide desired output but with less than optimum efficiency and the difference between optimum throughput and actual throughput increases exponentially. It takes long time to see the effects of Technical Debt. Therefore, testing is very less effective in discovering any Technical Debt present. Stress testing may provide some symptoms in some cases, however the difference between optimum and actual can be almost negligible in most of the cases.

Code review is a proactive way to settle a Technical Debt. Irrespective of the method used, if done properly, code review can identify potential Technical Debts at very early stage.

B. Reactive Technical Debt Settlement Process

Currently there is no established process of any kind to settle the Technical Debts. Technical Debts settlement process can be set as

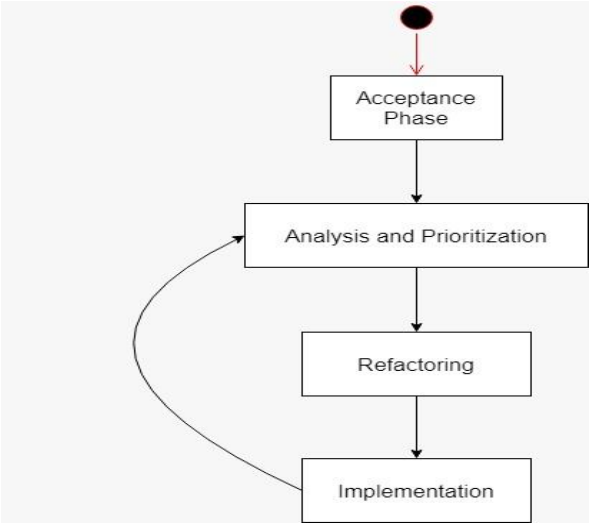


Figure 3 : Reactive Process for Technical Debt Settlement

1) Step 0 : Acceptance Phase : Most important thing is to accept that there are Technical Debts on the software. Without accepting the fact, it is impossible to go ahead.

Technical Debts are not only taken while design or coding. Any software, if not refactored for long time, itself generates KSTD or ECTD type Technical Debts. For example, if a code is untouched for very long time, mostly falls to vulnerabilities discovered meantime. Therefore, a code must be continuously analysed and enhanced.

Therefore, technical stakeholders must set a frequency and assign adequate resources for this activity

Business stakeholders also participate in taking new Technical Debts knowingly or unknowingly by not following Software Engineering practices. Every skipped Software Engineering step adds up a Technical Debt.

Irrespective of the role played, every person, who come across the software at any point of time in the total lifespan of the software, leaves a mark on the software.

Phase Deliverables -

a) RACI for Technical Debt Settlement process

b) Plan and process document

2) Step 1 : Analysis and Prioritization Phase: As, the team has accepted the fact that there is at least one Technical Debt on the software, in this phase, the team should identify, record, analyse and prioritize the Technical Debts.

Prioritizing of Technical Debts is to be completely based on the Software Quality Attributes and age.

The sequence of settling Technical Debt also depends on internal dependencies.

Phase Deliverables

a) List of identified Technical Debts

Impact estimation

Dependencies involved if any

©IJRASET (UGC Approved Journal): All Rights are Reserved

b) 80 – 20 analysis

Success criteria for each selected Technical Debt for current cycle

3) *Step 2: Refactoring Phase:* Design the solution keeping all the identified Technical Debts in mind.

Phase Deliverables

a) Updated design documents

4) *Step 3 : Implementation Phase:* This phase involves –

a) Coding
b) Code review
c) Testing phases
d) Release

Test Driven Development should be preferred to improve effectiveness of this process and to avoid breaking old running codes.

Phase Deliverables

Testing evidences

Continue to next cycle from Step 1

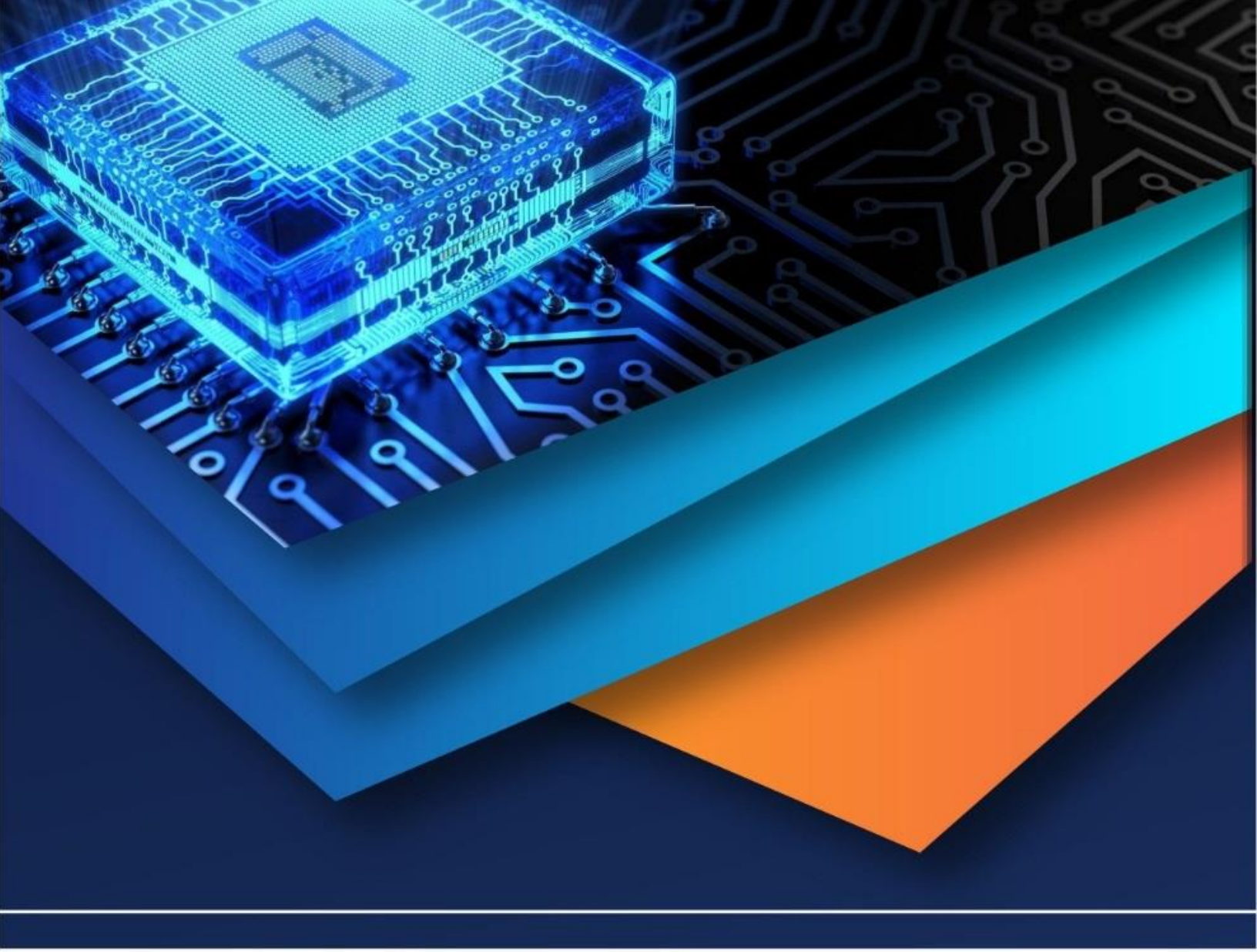
V. CONCLUSION

Not taking a single Technical Debt through out the life of a software is an impossible dream. In this author's opinion, taking a Technical Debt is not an issue, it is unavoidable in most of the cases. Not settling them is the real problem.

Just like unsettled financial debts, unsettled Technical Debts can also cause a lot of trouble over the period. Older the debt, larger the impact. As Technical Debts are not defect or bug in most of the cases, settling the Technical Debts doesn't get adequate attention.

Just like human beings, software also goes through Birth, Learning, Earning phases in its life. A software stays in Earning phase only till the value generated by it is greater than its maintenance cost. That also means, longer the Earning phase, more successful the software is. For longer and healthy life, a software must not have any old unsettled Technical Debts. Therefore, software must go through ‘periodic checkups’ for any Technical Debts

©IJRASET (UGC Approved Journal): All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)