**Technical Debt**

It illustrates the cost of rework caused by implementation of an immediate simple solution rather than a better approach that could take longer

Just Like financial debt, technical debt also is comprised of two elements.

1. The debt itself (initial design flaws)
2. The interest associated with the debt (extra effort that goes in to change the original code because of complexity of initial debt in the first step

**Types of technical debts:** Technical debt can be of 2 types

**Intentional** – team members know it and is well documented to take it up in the future

**Benefits of intentional technical debt:** Sometimes the technical debt can be useful in the following ways

* **Meeting time-to-market deadline**: If the project timelines are too tight then if we must release the product on time, we can have some technical debt which can be taken up by the team after the release (things that won’t affect the release) and concentrating on things that are needed for release itself
* **Creating POCs and MVPs:** We can also have technical debt when we create Proof of concept (POC) or Minimum viable product (MVP) to quickly show customer something. Here technical debt won’t hurt as its not very important how implemented the POC and can save us the time needed to implement the POC

**Non-intentional** – Here the team doesn’t know the impact of the tech debt. This is dangerous.

**Reasons for Non-Intentional tech debt**: All of the below items lead to a poor initial design flaws which will lead to tech debt eventually

* Lack of collaboration within the team
* Lack of experience in the team
* Poor documentation

**Technical Debt Indicators**

To reduce the tech debt, we need to look out at 3 areas. delivery, architecture, and people. One of them could be the reasons for technical debt and sometimes they might not be alone the reason for it.

**Delivery Indicators:**  Following are few aspects using which delivery managers can recognize the technical debts

* **Quality Degradation:** As technical debt increases the team experiences increase in the number of
  + ***Production issues*:** (Defects and production bugs).
  + ***Regression issues*:** There might happen that if the code is changed in some part of the system, the defect may arise in other place due to existing technical debt
  + ***Known issues*:** The issue that team are aware of and not important for release. A list of known issues itself indicate the tech debt that team has
* **Hight cost of system change:** if the time and effort taken is high to make a change in the system then it’s a sign of technical debt
* **Inability to Experiment Quickly:** Any software development process involves preparing Spikes and POCs. These items should not take a lot of time usually and if we notice that working on these items is associated with problems like deployment, readability etc. then it’s an indication of technical debt
* **Increased Barrier to Entry:** If it requires expert knowledge to go through the code and only one or two teammates can implement the functionality (bus factor alert – Talks about number of people who leave the project, and the project gets halt) and learning curve is high for a new joiner then it’s a tech debt

**Architectural Indicators:** Following are few things that can show us the technical debt

* **Hard to Integrate:**  Team finds it difficult to integrate the new features into the system
* **Hard to Ruse:**  A system should have **high cohesion** (Elements within one class/module should functionally belong together and do one thing.) and **low coupling** and if it’s the reverse then it’s a sign of tech debt
* **Hard to Grow:** If its hard to increase the performance
* **Hard to support:** Poor documentation quality and low system maintainability

**Team Indicators:**  Following are few factors related to teams to recognize the tech debt

* **Low Confidence in Scope Estimation:** If the team under estimates or over estimates the project scope. Its hard for team to male proper estimations because of unknows issues that can be caused by tech debt
* **Demotivation:** Impact of tech debt can demotivate the team requiring them to code quickly without following best practices

**Tracking technical debt:**

* **Keeping tech debt Registry up to date:** It’s a simple shared document where team puts all the tech debts and start looking whenever feasible
* **Using tech debt backlog:** List of items to be addressed by prioritizing them
* **Using tech debt management tools:** Unlike the above two where team needs to update, the tech debt management tool automatically identifies tech debts Tools like Sonar Qube themselves helps us to visualize and track
  + The % of test coverage
  + The code complexity index
  + The number of rule violations

**Delivery.epam.com** is used by managers across EPAM to track their projects

**Managing tech debt:** Managers and leaders often use RAG (Red, Amber, Green) colors to report the status of the project

Red – There are issue in the team that need to be fixed for proper project delivery

Amber – It implies that project has some potentials issues and team might need to re visit in the future

Green – Project is healthy

**Recommendations for leaders:**

* Reserve time for resolving technical debt in a sprint or have a complete sprint if needed
* Motivate the team and explain the consequence of having tech debt in long term
* Make the data more visible to the team
* Let the customer/stake holders know about the tech debts and have a plan to mitigate the tech debt by including them in the backlog

**Recommendation for developers:**

* Avoid code smells by following clean code principles, SOLID principles and DRY
* Conduct code reviews
* Share knowledge
* Use code Analysis tools
* Use automation testing