SOFTWARE MAINTENANCE ASSIGNMENT 1

- ASHISH KUMAR
- 2K18/SE/041

Ques: For each of Lehman's laws provided, give one appropriate situation where it will be applicable in the context of Software Engineering.

Ans: **Lehman's laws of software evolution**

Software evolution is a concept in software engineering that means that software is developed and over time undergoes iterative changes.

There have been many attempts in the past to understand exactly how software systems evolve. One of such attempts is the one made by **Manny Lehman** and his colleague **Lehman** in 1974. They studied and identified a set of behaviors that later became known as **Lehman's Laws of Software Evolution**. These laws describe a balance between forces driving new developments on one hand, and forces that slow down progress on the other hand.

The 8 laws that were formulated are:

1. "**Law of Continuing Change**" — A system must be continually adapted or it becomes progressively less satisfactory. The first law states a program that is used in a real-world environment must necessarily change or else become progressively less useful in that environment.

   Example: As time goes technology changes and it becomes necessary for the company to improve existing system. Like Android gives security update every month in smartphones so that mobiles can prevent from any phishing attack from hackers or crackers as the risk of hacking is increasing day by day, so tech companies must also be updated with these types of attacks.

2. "**Law of Increasing Complexity**" — As a system evolves, its complexity increases unless work is done to maintain or reduce it. The second law states that, as a system is changed, its structure tends to become more complex. The more functionality added in each release of software, more complexity will be there.

Example: As time goes, the software is also meant to evolve as more and more features are included in it as requirements changes over a period of time. Like Paytm app, when it was launched, it simply had some payment features. But now as new features added you can buy gold from it, can book movie tickets, can take insurance and many more things which increase the complexity of app.

3. "**Law of Self Regulation**" — A system evolution processes are self-regulating with the distribution of product and process measures close to normal.

4. **"Law of Conservation of Organisational Stability** (invariant work rate)" — the average work rate in an E-type process tends to remain constant over periods of system evolution. Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.

5. "**Law of Conservation of Familiarity**"—the average incremental growth remains invariant as the system evolves. Once the software has been released, its changes in each release would be constant, and unless environmental factors change drastically, the changes would be familiar. Example: Adding new functionality to a system inevitably introduces new system faults. The more functionality added in each release, the more faults there will be. Therefore, a large increment in functionality in one system release means that this will have to be followed by a further release in which the new system faults are repaired. Relatively little new functionality should be included in this release. This law suggests that you should not budget for large functionality increments in each release without taking into account the need for fault repair.

6. "**Law of Continuing Growth**" — the functional content of a system must be continually increased to maintain user satisfaction over its lifetime. The sixth law states that the functionality offered by the systems has to continually increase to maintain user satisfaction. Example: Let's take an example of Instagram. Before Tiktok's ban there was no feature of reels in it and after ban of tiktok many users starts using reels feature of it, which increases

customers of Instagram. So this is a clear example of why functionality of software needs to be updated continuously.

7. "**Law of Declining Quality**" — the quality of a system will appear to be declining unless it is rigorously maintained and adapted to operational environment changes. Example: Suppose, a system is made to handle 1000 users at a time and now if there is increasement of users and the system is not able to handle it, then system will crash which indicates that the quality of software is poor, so the system needs to update regularly to maintain the same quality.

8. "**Law of Feedback System**"— evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base. Example: Many companies releases there softwares in beta stage before launching globally so that they can take feedback from beta testers public and can improve the software.