Lehman's Law

The Lehman laws describe a balance between forces driving new developments on one hand, and forces that slow down progress on the other hand.

Lehman distinguishes between three categories of software:

- A S-program (specified) is written according to an exact specification
- A P-program (procedural) is written to implement certain procedures that completely determine what the program can do
- An E-program (evolving) is written to perform some real-world activity, such a program needs to adapt to varying requirements and circumstances in that environment

The laws are said to apply only to the last category of systems

Continuing Change

An E-type system must be continually adapted or it becomes progressively less satisfactory

Increasing Complexity

As an E-type system evolves, its complexity increases unless work is done to maintain or reduce it.

Self Regulation

E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.

Conservation of Organisational Stability (invariant work rate)

The average effective global activity rate in an evolving E-type system is invariant over the product's lifetime.

Conservation of Familiarity

As an E-type system evolves, the average incremental growth remains invariant as the system evolves.

Continuing Growth

The functional content of an E-type system must be continually increased to maintain user satisfaction over its lifetime.

Declining Quality

The quality of an E-type system will appear to be declining unless it is rigorously maintained and adapted to operational environment changes.

Feedback System

E-type 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.

Code Salvaging

Code salvaging assigns the new developer the task of opening thousands of lines of code (depending on the size of your project), reverse engineering the logic of another developer’s logic, rewriting any efficiencies while converting incompatibilities.