Leyes de Lehman

Fausto Panello 2025

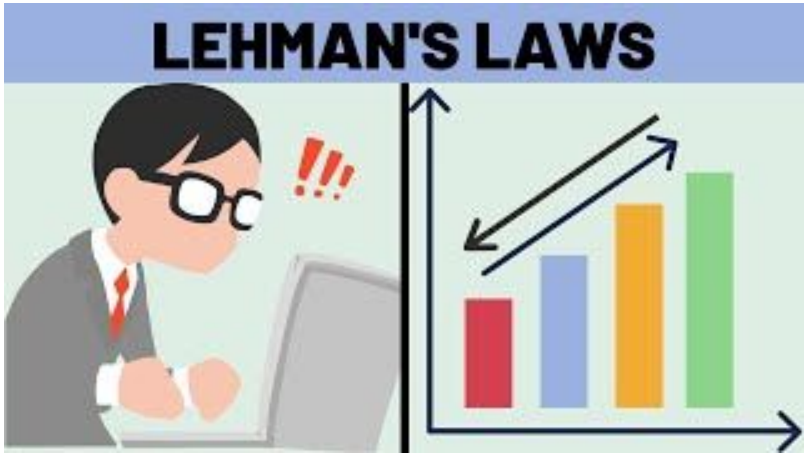
Meir M. Lehman

Meir Lehman era un profesor en la Escuela de Ciencias Computacionales de Middlesex, Londres.

Estudió matemática en el Colegio Imperial de Londres, y en los 70s se fue a trabajar a Nueva York, en la división de descubrimiento de IBM.

El estudio del proceso de programación de IBM, dió las fundaciones para que Lehman desarrolle las leyes de evolución de software, a principios de la década de los 80.

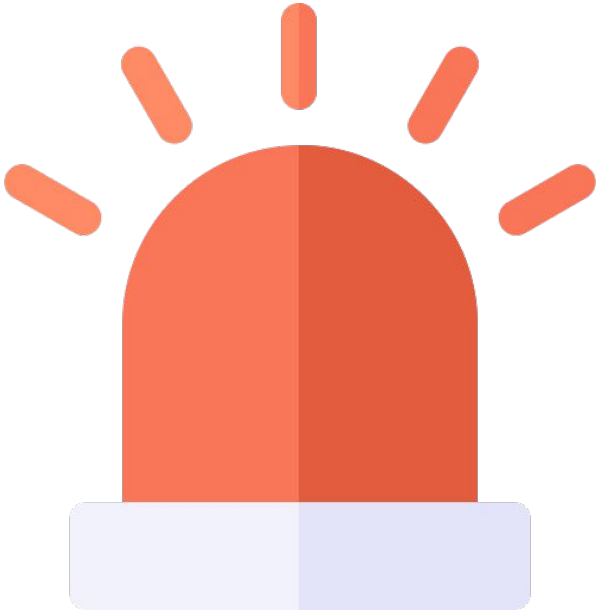
¿Por qué diseñó leyes?



A principios de los 80, justo cuando Lehman entró a IBM, estaban en pleno auge las metodologías ágiles (espiral, incremental, etc.).

Es así que Lehman observó que se configuraba una nueva problemática en el desarrollo de sistemas, es decir, la crisis se expande ahora en el sentido que no sólo se requiere reflejar lo más fielmente posible las necesidades del usuario, sino que ahora los ambientes en que el sistema está inserto están sujetos a cambios y estos cambios inciden en la efectividad del software desarrollado.

¿Por qué diseñó leyes?



A su vez, ocurre un fenómeno: la **crisis del software**. Esto se refiere a los problemas que, desde sus inicios, ha ido experimentando el software, muchas veces problemas de gran magnitud, debido, principalmente, a la mínima eficacia que presentan una gran cantidad de empresas al momento de realizar un software; sería algo así como el conjunto de dificultades o errores ocurridos en la planificación, estimación de los costos, productividad y calidad de un software.

Sin embargo, la crisis del software no causaba solamente problemas económicos…

Ej. Crisis del software

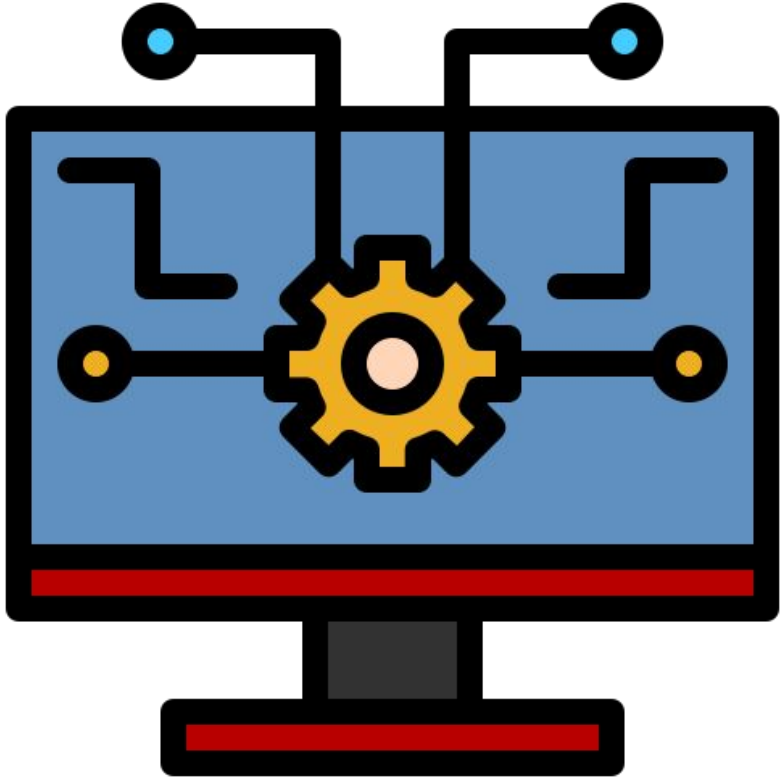
1986: En abril de 1986 un avión de combate se estrelló por culpa de un giro descontrolado atribuido a una expresión “if then”, para la cual no había una expresión “else”, debido a que los desarrolladores del software lo consideraron innecesario.

1985-1987: El Therac-25 fue una máquina de radioterapia que causó la muerte de varios pacientes en diversos hospitales de Estados Unidos y Canadá, debido a las radiaciones de alto poder aplicadas sin control, las cuales fueron atribuidas a la falta de control de calidad del software médico.

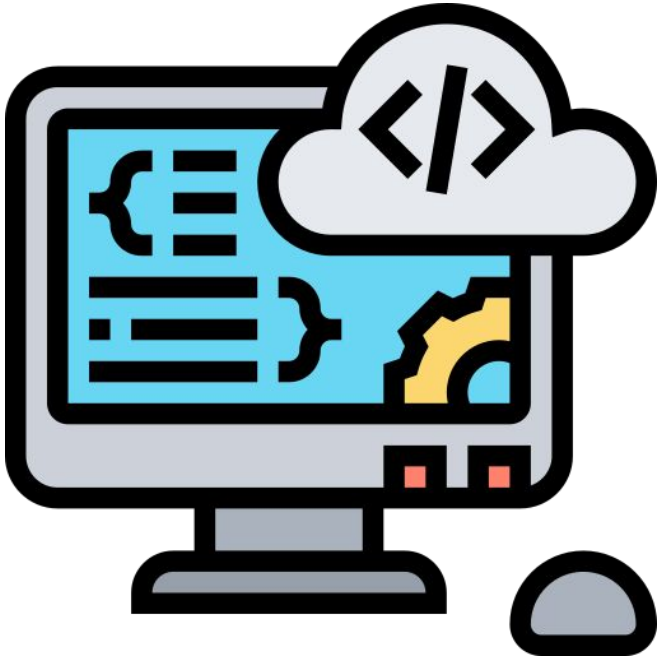
Leyes de Lehman

Lehman propone que la evolución de un sistema de software está sujeta a varias leyes. Ha determinado estas leyes a partir de observaciones experimentales de varios sistemas, como los grandes sistemas operativos.

Lehman señala, a partir de esto, 3 tipos de programas, y varias leyes.



‘Programas’ de Lehman

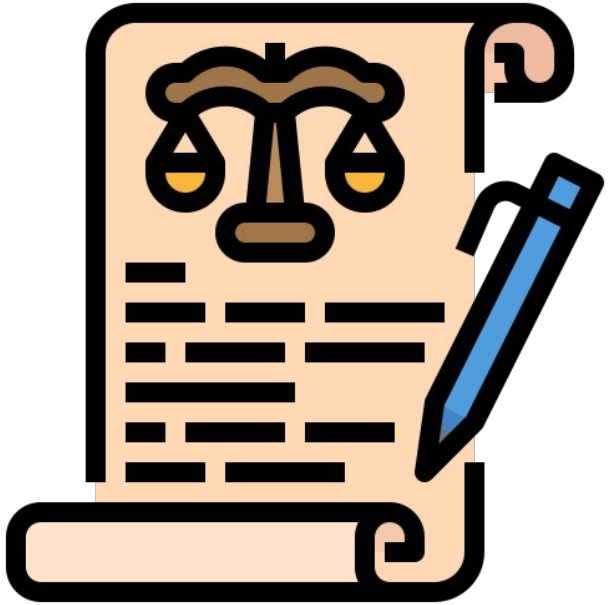


Un **‘S-Program’** es escrito de acuerdo a una especificación exacta de qué puede hacer el programa. Por ejemplo, un programa que consista en un algoritmo determinado, estático y sin evolución; el problema de las 8 damas.

Un **‘P-Program’** es creado para implementar ciertos procedimientos que determinan completamente qué puede hacer el programa de acuerdo al contexto; un motor de ajedrez.

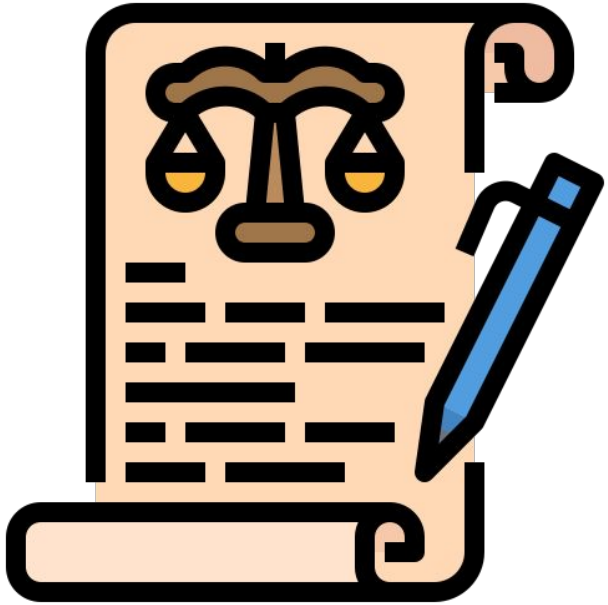
Un **‘E-Program’** se crea para hacer actividades de la vida real; cómo debe comportarse está fuertemente vinculado al entorno en el que corre, y semejante programa necesita adaptarse a requerimientos variados y circunstancias en ese entorno.

Leyes de Lehman



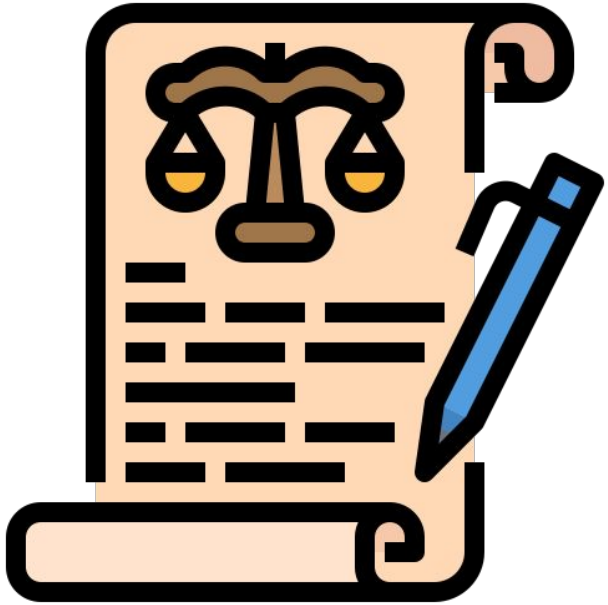
- **Cambio continuo:** Un programa que se utiliza en un ambiente del mundo real debe cambiar o será cada vez menos útil en ese ambiente.
- **Complejidad creciente:** A medida que un programa en evolución cambia, su estructura se hace más compleja, a menos que se lleven a cabo esfuerzos activos para evitar este fenómeno.
- **Evolución del programa:** La evolución del programa es un proceso autorregulado, y una medición de atributos del sistema, como el tamaño, el tiempo entre versiones, el número de errores advertidos, etc., revela las tendencias estadísticas significativas y las características invariantes.

Leyes de Lehman



- **Conservación de la estabilidad organizativa:** Durante el tiempo de vida de un programa, su rapidez de desarrollo es casi constante e independiente de los recursos dedicados al desarrollo del sistema.
- **Conservación de la familiaridad:** A medida que evoluciona el sistema, todo lo asociado con ello, desarrolladores, personal de ventas y usuarios, por ejemplo, deben mantener maestría de su contenido y comportamiento para lograr la evolución satisfactoria y constante. El crecimiento excesivo disminuye esta maestría. Entonces cada incremento de los incrementos debe mantenerse promedio e invariante.

Leyes de Lehman



- **Crecimiento continuo:** El contenido funcional de un sistema debe ser incrementado continuamente para mantener la satisfacción de los usuarios y clientes a lo largo de su ciclo de vida.
- **Calidad declinante:** La caliad de un sistema va a ser aparentemente declinante salvo que se mantenga y se adapte rigurosamente para acatar los cambios operacionales ambientales.
- **Sistema de retroalimentación:** El proceso de evolución de un sistema constituye sistemas de retroalimentación multi-nivel, multi-iterativos, y multi-agentes, y deben ser tratados para denotar una mejoría significativa.

Interpretaciones

Somerville: Considera las "leyes" de Lehman como elementos positivos en la configuración del ámbito en que se desarrollan los sistemas, por ejemplo, dice que las dos primeras hipótesis de Lehman son casi con certeza válidas.
Interpretando la primera como: “...El razonamiento subyacente en la primera ley... es que cuando un sistema de software (sobre todo uno grande) se construye para modelar algún ambiente y después se introduce en él, modificándose así el ambiente original con la presencia del sistema de software... No puede ser más interesante para los propósitos de esta revisión la característica planteada por esta hipótesis, es decir, si se desarrolla software no se podría pensar desde la perspectiva de estabilidades, todo lo contrario, necesariamente es el cambio el motor y el configurador de los sistemas de software.”

Sommerville

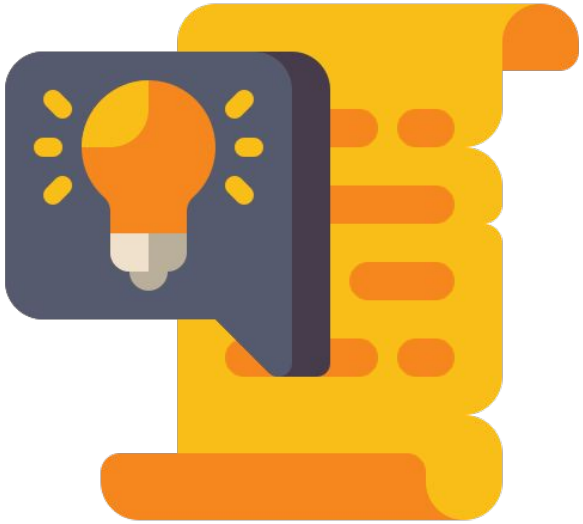
La segunda ley en el análisis de Sommerville corresponde al hecho de que la estructura del programa original se estableció para aplicarlo a un conjunto de necesidades iniciales. A medida que se produce el cambio evolutivo de esas necesidades, la estructura original se degrada y con ello se va haciendo más compleja ya que al ir incrementándose la distancia conceptual entre el software y el medio que lo contiene, éste va estando cada vez más presente en el hacer cotidiano, formulando quiebres constantes sobre el sistema de trabajo lo que imprime un sello de complejidad a la utilización del software.

Sommerville

Las siguientes leyes tienen relación con las características de las organizaciones y de los individuos que participan en el proceso de desarrollo de software. Lehman afirma que las organizaciones se esfuerzan por lograr la estabilidad e intentan evitar cambios drásticos o repentinos.

Por lo tanto, a medida que se añaden más recursos a un proyecto de software, el efecto evolutivo de la adición de nuevos recursos se va reduciendo, hasta que la adición de nuevos recursos no produce ningún efecto. Si bien, estas últimas leyes no resultan tan obvias como las primeras y podrían ser cuestionadas, es posible que la última, que tiene que ver con la conservación de la familiaridad sea la más útil, como también la de reducción de personal, en el sentido que cuanta más gente trabaje, menos productivo será cada miembro del proyecto.

Richard Fairley



El desarrollo de productos mediante el método de versiones sucesivas es una extensión del método de prototipos en el que se refina un esqueleto inicial del producto obteniendo así, cada vez más capacidades. En dicho método, cada versión es un sistema funcional y capaz de realizar trabajo útil.

En realidad, el ciclo de desarrollo de un producto de programación es una combinación de los distintos modelos presentados. Las organizaciones y proyectos especiales pueden adoptar alguno de estos modelos en particular; sin embargo, ciertos elementos de ellos se encuentran en todo proyecto de programación. Por ejemplo, para proyectos de desarrollo de programación no es extraño adoptar el modelo de fases como marco de referencia básico, e incluir prototipos y versiones sucesivas en el desarrollo.