Alumno: Kevin Efrén Salas Martínez.

Profesor: Eduardo Gallegos Flores.

Materia: Ingeniería de Software. Unidad III TAREAS

Fecha: 1/04/2019

Carrera: Ing. Tecnología de la información y comunicación (TIC´s), 4° semestre.

Reseña de Leyes Famosas Del Desarrollo De Software.

Empecemos hablar sobre las famosas leyes del desarrollo de software, como bien decimos en este mundo del desarrollo del software se cuenta con su propias reglas, principios y leyes interesante y muy conocidas por los programadores.

Uno como programador, diseñador, gerente y arquitecto a menudo usa sin que nos demos cuenta ejemplo en las conversaciones, reuniones y chats ente muchos más ejemplos.

Estas leyes o reglas consisten en principios o palabras famosas de personas grandes e inspiradoras en el mundo del desarrollo.

Así mismo a lo largo del tiempo son muy interesantes, divertidas y realmente vale la pena conocerlas.

En este documento se darán a conocer las interpretaciones y pensamientos sobre las leyes más famosas y más utilizadas para el desarrollo del software.

A continuación se darán a conocer estas famosas leyes así como su descripción:

- Ley de murphy:

  Esta ley es de las más conocidas, sobre todo porque no solo aplica al desarrollo del software. (Si algo puede salir mal, lo hará.)

- Ley de Brook:

  La gran mayoría de desarrolladores, ya sea con conocimiento o sin conocimiento, un tendrá experiencia con la ley de Brook. Que nos dice; La adición de mano de obra a un proyecto de software tardío lo hace más tarde.

- Ley de Hofstadter

Esta ley fu escrita por Douglas hofstadter y lleva su nombre, esta ley nos habla sobre la dificultad de estimar con precisión el tiempo que tomara uno parar completar la complejidad sustancial.

- Ley de Conway

  Las organizaciones cuyos sistemas de diseño están obligados a producir diseños que sean copias a las estructuras de comunicación de esta organización.

- Ley de Postel

  Sean conservador en lo que envían, sea liberal en lo que aceptas.

  Esta ley es un ejemplo de unificación.

- Principio de Pareto

  Para muchos fenómenos, el 80% de las consecuencias provienen del 20% de las causas.

- El principio de peter

  Esta ley es bastante deprimente y a veces frustrante.

  En una jerarquía, cada empleado tiende a elevarse a su nivel de incompetencia.

- Principio de Kerchkhoff

  El principio fundamental de esta ley es la criptografía de clave pública.

- La ley de linus

  Esta ley fue nombrada a honro de linus torvald, el creador de Linux y establece que los dados suficientes globos oculares, todo los errores son superficiales.

- Ley de Moore

  El número de transacciones en un circuito integrado se duplicaran en aproximadamente 18 meses.

- Ley de wirth

  El software se ralentiza más rápido que el hardware.

- Regla del noventa y noventa

- Principio de optimización de Knuth

  El primer 90% del código toma el 10% del tiempo. El 10% restante toma el otro 90% del tiempo.

- La ley de norvig

  La optimización prematura es la fuente de todos los males, primero se escribe el código, luego identifican los cuellos de botella y luego solución de los problemas.

Bibliografía

timsommer. (18 de diciembre de 2017). Recuperado el 9 de Abril de 2019, de https://www.timsommer.be/famous-laws-of-software-development/

Como ha evolucionado la ingeniería de software hace 50 años.

Viajemos hace 50 años cuando fue el nacimiento de la ingeniería de software en el ámbito de disciplina, nadie se imaginó tener un gran impacto en esta actualidad.

Recuerdo la primera computadora en las instalaciones de la UNAM hace 60 años.

Se le reconoce a Margaret Hamilton, ya que el dirigió el desarrollo del sistema de navegación del sistema espacial el muy conocido Apolo por acuñado en 1965 el termino la ingeniería de software.

Posteriormente también termino con colaboradores como Anthony Oettiger (presidente del ACM) en 1966, y Friedrich Ludwic Bauer del comité de ciencias de la OTAN en 1967.

El nacimiento de esta disciplina ocurrió en octubre de 1968, cuando el comité de ciencias de la OTAN, sufría de una crisis de software, para la cual convoco a los mejores científicos en computación.

Los problemas que sufrieron en 1968 fueron una falta de compresión del proceso de diseño del programa.

La construcción del sistema.

Esto se logró hasta el 2018, durante los últimos 500 años han trascurrido diferentes modelados de software tales como; cascada, espiral, iterativo-incremental o ágil tipo Scrum.

Se han generado el manifiesto de desarrollo de ágil software, que esta combinando la forma de trabajar y las relaciones humanas.

Para la creación de la fiabilidad de un software en aquellos tiempos era suficiente en los sistemas, cada vez eran más integrados en las actividades centrales de la sociedad.

La calidad de software cuenta con dos versiones; una es la del proceso y la otra del producto. En la primera introduce conceptos y técnicas, y en la segunda asegura la calidad y la mejora de los procesos.

El costo del desarrollo del software estaba integrado con los costos de desarrollo de hardware. No se cuantificaba el valor y costo del software como tal.

Para justificar el costo del software se tuvo que empezar<r a medir el tamaño del software, en el primer intento se basaron en las líneas de código (LOCs), lo cual fue criticado porque el software funciona igual pero con diferentes tamaños de LOCs. Luego estuvo en propuesta ajustar unos ciertos puntos de función o parámetros de complejidad del software a los cuales se les llamaría “Puntos Funcionales Cosmic”.

El problema de la intangibilidad del software sigue causando estragos en la estimación de costos.

A pesar de los avances, se ha visto que la gerencia de programación continuara mereciendo su mala reputación.

Para mejorar estos aspectos se ha creado la carrera de Ingenieras de software, la primera a nivel mundial fue fundad en 1996 en Rochester Institute of Technology por un profesor mexicano.

En el mercado laboral es relativamente inmaduro y prefiere las certificaciones de alguna habilidad técnica que profesional.

Los códigos de ética profesional de los desarrolladores de software tampoco han tenido gran difusión.

Bibliografía

sg. (26 de octubre de 2018). Recuperado el 16 de abril de 2019, de https://sg.com.mx/revista/58/50-anos-de-la-ingenieria-de-software-problemas-logros-tendencias-y-retos