RESEÑAS DE LECTURAS

Jaqueline Garcia Luevano

Eduardo Flores Gallegos

RESEÑAS DE LAS LECTURAS

LECTURA 1:

11 razones por las que el desarrollo de software profesional nunca dejará de existir

En la lectura “11 razones por las que el desarrollo de software profesional nunca dejará de existir” da respuesta, a mi parecer, a una de las preguntas más frecuentes en el entorno de los desarrolladores ¿el desarrollo de software será siempre demandado por el mercado laboral?, la respuesta es sencilla, sí lo será. La tecnología avanza a pasos agigantados y con eso logra abrir un sinfín de oportunidades laborales, así como adecuar las existentes ya que en estos tiempos lo que no avanza, queda obsoleto. Cada vez aparecen más y nuevas herramientas para desarrollar y detrás de ellas hay programadores los cuales hacen realidad su existencia. El desarrollo de software no solo indica código si no que atiende las problemáticas y da una solución a ellas en cuanto a software se habla. Como bien se menciona, los desarrolladores de software siempre van a ser relevantes ya que estamos en una época en el que términos como tecnología, software y programación son encontrados en todas partes además de que a pesar de que la mayoría de la población ya interactúa con la tecnológica no quiere decir que estos se vayan a enfocar en el desarrollo de software. Otro punto es, que el software es necesario en todas las empresas, en lo personal veo casi imposible que una empresa que se dedica a hacer grandes producciones de X cosa, no tenga un software el cual ayude al manejo de esto y detrás de ello un ingeniero en software y/o un programador. En general los puntos de esta lectura logran aclarar el por qué el desarrollo de software y a su vez los desarrolladores son y serán un factor elemental en la sociedad laboral de hoy en día, y con ello logra a motivar a aquellos que apenas están comenzando.

LECTURA 2:

Importancia del Software

Esta lectura logra tocar puntos relevantes en cuanto a la importancia del software, destaca que este debe de tener calidad para obtener como resultado un buen funcionamiento, idea la cual comparto, ya que si este no presenta calidad quiere decir que no se desarrolló con dedicación o que hubo fallas a lo largo de su elaboración. Hoy en día la ingeniería de software toma poder ya que gracias a esta podemos evaluar cada fase del desarrollo de un programa. Si no existiera esta ingeniería los procesos se verían truncados y no llevarían un orden en específico dando como resultado un software de mala calidad o uno que no cumple con las especificaciones requeridas. Hoy en día el software es muy esencial ya que este nos permite hacer actividades de todo tipo, desde escribir un documento hasta proteger nuestra computadora. Hablando de tecnología, software, avances y procesos, ¿Qué hay de la sociedad que no cuenta con estos conocimientos? ¿Qué se hace al respecto para evitar las brechas digitales?, el autor de esta lectura destaca algunos posibles motivos como: que no hay una buena formación en la sociedad con respecto a esto, que el nivel socioeconómico influye y/o que hay sociedades que se creen desarrolladas por contar con muchas máquinas pero que no tienen un conocimiento compartido, yo agregaría factores como la política, ya que en algunos países no se tiene la libertad de información y opinión, agregaría también los factores lingüísticos ya que no todas las personas conocen siquiera lo básico del idioma ingles y como lo sabemos casi toda la información más actualizada esta en este idioma.

LECTURA 3:

Leyes famosas de desarrollo de software

Esta lectura da a conocer algunas leyes en el entorno de desarrollo de software (desconocidas para mi) las cuales fueron impuestas por personas importantes en el desarrollo del software. Las que más llamaron mi atención fueron la ley de Murphy ya que concluye que una computadora hará lo que escribes, no lo que quieres y en concreto el programador tiene que lidiar con esa característica ya que tiene que hacer que lo que escriba sea entendible para la máquina y se pueda ejecutar lo deseado. La ley de Brook la cual menciona que agregar a personas en una parte avanzada del proyecto para hacerlo más rápido puede ser contraproducente ya que ocasionaría retrasos, esto es debido a que el nuevo integrante tiene que comprender todo lo relacionado al proyecto y luego ver la posible solución. Ley de Hofstadter la cual nos dice que siempre lleva más tiempo de lo esperado desarrollar y en efecto porque hasta cierto punto siempre va a faltar tiempo. La ley de noventa y noventa fue una que me sorprendió debido a que esta dice que el primer 90% del código toma el 10% del tiempo y el 10% restante toma el otro 90% del tiempo, esto porque pueden surgir complicaciones imaginables que consumen el tiempo. Principio de robustez de la Ley de Postel el cual nos dice: sea conservador en lo que envía, sea liberal en lo que acepta, lo cual tiene sentido ya que al momento de desarrollar es válido recibir información de tantas fuentes como sean necesarias. Sin duda todas las leyes descritas tienen su toque divertido, pero totalmente apegado a la realidad.

LECTURA 4:

50 Años de la Ingeniería de Software Problemas, Logros, Tendencias y Retos

Así como todo tiene historia, la ingeniería de software no es la excepción, a principios de su desarrollo tuvo complicaciones y estas predominaban en que hacía falta una comprensión más completa del proceso de diseño. Al paso de los años ha ido evolucionando de tal manera que ya ha pasado por diferentes modelos. Si de calidad se habla esta también ha ido evolucionando y de buena manera ya que anteriormente no se tenía tanta fiabilidad en el software, todo se basaba en que funcionara. El autor menciona que, si nos enfocáramos más en la prevención de defectos en vez de los ciclos de pruebas y sus correcciones, la calidad del software aumentaría y en efecto, comparto la idea del autor ya que esto a su vez ahorraría tiempo y esfuerzo. Por otro lado, están los costos, un tema muy controversial desde el comienzo de desarrollo de software, ya que, hasta la actualidad muchos se cuestionan si la cantidad pagada es la justa y esto a pesar de que se tiene estándares como el estándar ISO/IEC 14143, el cual mide el tamaño funcional. En cuanto a la gestión del desarrollo de software, anteriormente era inestable ya que a pesar de tener una fecha limite por lo general no se cumplía, ya que era difícil calcular el avance del proyecto, pero gracias a que se disciplinó esta área ahora podemos ver proyectos si no completamente terminados, con un poco por terminar en la fecha indicada. Como todo, creo que es cuestión en la época en la que se vive ya que en los años de 1968 cuando apenas surgían estos nuevos conceptos en la sociedad no eran vistos como algo que fuera viable, sin embargo, el avance de la tecnología y todo lo que arrastró consigo implica que hoy en día la ingeniería de software sea una carrera que se pueda ejercer profesionalmente. Así mismo como desarrollador es importante tener siempre información actualizada, ya que, la tecnología crece a pasos agigantados y si no se asumen retos nunca se tendrá la mentalidad de innovar.

LECTURA 5:

Metodologías tradicionales VS. Metodologías ágiles

A lo largo de esta lectura se nos muestran diferentes metodologías al momento de desarrollar software, por un lado, están las tradicionales las cuales se basan en la planificación predictiva, con mucha documentación, desarrollo individual, es decir que haya una persona para cada rol en específico y a las actividades de control en las pruebas para así tener, valga la redundancia, un control en el proceso de desarrollo cumpliendo así con los objetivos finales. Un ejemplo es el proceso racional unificado, RUP por sus siglas en inglés, el cual al ser un método tradicional se basa en asignar tareas y cumplir con el cronograma con el fin de que se entreguen en tiempo y forma a los usuarios. Otro método tradicional es el Microsoft Solution Framework (MSF) el cual nos dice que cualquier proyecto se va a dividir en las etapas de visión y alcances, planificación, desarrollo, estabilización e implantación, lo cual tiene sentido ya que con esas etapas de desarrollo se puede llevar un control estricto. Dejando de lado las metodologías tradicionales tenemos las metodologías ágiles las cuales nos explican que es preferible dedicarse a la programación del producto en lugar de hacer una documentación que lleve mucho tiempo, estas siguen también la idea de que es mejor adaptar que seguir con un estricto plan ya que esto abre las puertas al cambio si es que es necesario. En esta metodología tenemos extreme programming la cual se basa en la adaptación, pruebas continuas, programación por parejas y fuerte interacción. Ambas metodologías son respetables tienen sus ventajas y desventajas, y en lo personal creo que los desarrolladores deben elegir aquella que se adapte al proyecto, a ellos mismos (ya que en un apartado de la lectura mencionan que es recomendable que aquellos que tengan experiencia con las metodologías tradicionales son los que podrían adaptarse a las agiles ya que requiere de estrategia) y a las necesidades del usuario.

BIBLIOGRAFÍA:

Fred (2017). Velneo. Recuperado de: https://velneo.es/11-razones-las-desarrollo-software-profesional-nunca-dejara-existir/

S.A (2011). Wordpress. Recuperado de: https://isoftwareunesum.wordpress.com/2011/04/28/importancia-del-software/

Sommer Tim. (2017). Recuperado de: https://www.timsommer.be/famous-laws-of-software-development/

Oktaba Hanna. (S.A). SG. Recuperado de: https://sg.com.mx/revista/58/50-anos-de-la-ingenieria-de-software-problemas-logros-tendencias-y-retos

Figueroa Roberth G., Solís Camilo J. ,. Cabrer Armando A. (SA). Recuperado de: https://trello-attachments.s3.amazonaws.com/5e1647e1e290f274a873acd2/5e39ac571df86f8ad47021d5/2f6fe5a3947912c8b468b657ce63952a/articulo-metodologia-de-sw-formato.pdf