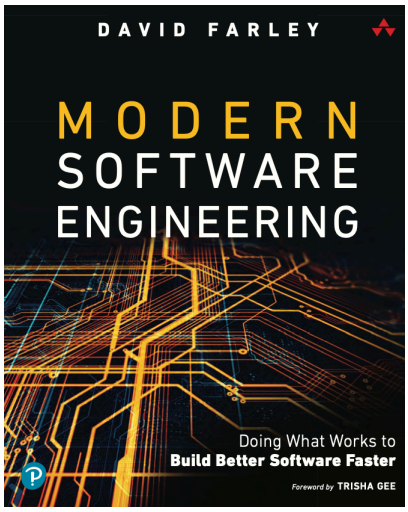
Blog de Dave Farley
Reflexiones sobre la entrega continua y el desarrollo ágil.

¿Qué es la ingeniería de software moderna?

Publicado el 30 de enero de 2022 por davef

He publicado un nuevo libro titulado “ Ingeniería de software moderna ” y he estado trabajando en él durante los últimos años.



Las ideas contenidas en él surgieron de una creciente comprensión de que la forma en que abordo el desarrollo de software y la forma en que todos los equipos con los que estaba familiarizado, que consideraba excelentes en el desarrollo de software, compartían algunas características fundamentales.

Esto me hizo interesarme en tratar de precisar esas características y formularlas en un modelo que pudiera usar para explicar lo que a mí me parecía que funcionaba mejor para el desarrollo de software.

Aplicando la ciencia

Me di cuenta de que mi propio pensamiento ha estado influenciado durante mucho tiempo por una de mis aficiones: me gusta leer y aprender sobre ciencia. No solo me interesan los hallazgos científicos, sino también el enfoque organizado de la adquisición de conocimientos que representa. La ciencia es el mejor enfoque de la humanidad para la resolución de problemas, por lo que sin duda debería ser aplicable a una disciplina técnica y difícil como el desarrollo de software.

Hace tiempo que describí mi enfoque preferido para el desarrollo de software, **la entrega continua, como una aplicación simple y pragmática del razonamiento de estilo científico para resolver problemas en el software** . Esto me llevó a interesarme mucho en explorar esta idea con más profundidad. ¿Qué significa aplicar un enfoque informal al aprendizaje y descubrimiento científicos cuando lo aplicamos a la resolución de problemas prácticos? Hay una palabra para eso, lo llamamos ***"Ingeniería"*** .

Ingeniería != Burocracia

En ese momento me puse un poco nervioso. Me parece que en nuestra disciplina del desarrollo de software el término “ingeniería” se ha cargado incorrectamente de significado o se ha vaciado por completo de él.

Por un lado, muchas personas asumen que “Ingeniería” significa sofocar la burocracia y controlar procesos pesados.

Por otro lado, “Ingeniería” significa simplemente escribir código y nada más.

Ambas cosas son profundamente erróneas. En otras disciplinas, **la “ingeniería” es simplemente lo que funciona** . Es práctica, pragmática y más eficiente (no menos).

Claro, puede ofrecer algunas pautas que limiten nuestro pensamiento, pero lo hace de una manera que nos ayuda a descartar, o al menos a alejarnos, de ideas tontas. ¡Y eso es algo realmente bueno!

Cómo evitar las malas ideas

En el desarrollo de software, no siempre hemos sido capaces de eliminar las malas ideas. Tendemos a cometer los mismos errores una y otra vez.

> *Los científicos de datos no utilizan el control de versiones, lo que da como resultado que el 90 % de los proyectos de ML nunca lleguen a producción* .
>
> *Entornos de bajo código que funcionan bajo el supuesto de que sabes exactamente lo que quieres al inicio de un proyecto y que ese requisito nunca cambiará: ¡buena suerte con esa idea!*

Por lo tanto, sería muy valioso tener algo que pudiera alejarnos de las malas ideas.

El error de los mil millones de dólares

La idea de que **“Ingeniería == Burocracia” es errónea** y proviene de una categorización errónea, completamente incorrecta, pero comprensible, de lo que es la ingeniería en otras disciplinas, y luego de aplicar esa categorización errónea al software.

Los seres humanos estamos acostumbrados a construir cosas físicas, por lo que la producción de cosas físicas ocupa un lugar central en nuestras mentes cuando pensamos en fabricar cosas. La inspiración y el diseño de una cosa física es sin duda un problema complejo, pero es mucho más difícil ampliarlo para producir esas cosas en masa, por lo que asumimos que ahí es donde reside el único desafío real y, por lo tanto, asumimos que eso es todo lo que hace la ingeniería.

Suponemos que “Ingeniería == Ingeniería de producción” y, por eso, como industria, cometimos el error de mil millones de dólares de intentar mejorar la eficiencia del desarrollo de software aplicando técnicas de línea de producción. ¡No funcionó!

La producción no es nuestro problema

En el software, la “producción” no es nuestro problema. Nuestro producto es una secuencia de bytes y podemos recrear cualquier secuencia de bytes prácticamente sin coste alguno.

Esto significa que **NUNCA tenemos un problema de producción** . Nuestro problema siempre es de aprendizaje, descubrimiento y diseño. Por lo tanto, la ingeniería de software debe centrarse firmemente en esa parte del desafío e ignorar, o al menos automatizar, nuestro proceso de producción.

Ingeniería de diseño NO ingeniería de producción

Entonces, ¿cómo optimizamos la exploración, el aprendizaje y el diseño?

Si queremos buscar ejemplos fuera del ámbito del software, esto está mucho más relacionado con la creación innovadora de cosas nuevas que con la ingeniería de producción. La ingeniería de diseño es una disciplina muy diferente. Pensemos en la NASA diseñando los exploradores de Marte, en Apple diseñando el primer iPhone o en SpaceX diseñando su Starship.

En este tipo de ingeniería, hay que optimizar el aprendizaje. La ingeniería moderna diseña conscientemente sistemas que permiten a los ingenieros iterar de forma rápida y eficiente para poder aprender qué funciona y qué no. Nosotros debemos hacer lo mismo.

Diseño de sistemas complejos

Los sistemas que crean los ingenieros modernos son cada vez más complejos y sofisticados, por lo que, además de centrarse en el aprendizaje, la ingeniería moderna en general, pero sin duda la ingeniería de software moderna, debe centrarse en la gestión de esa complejidad. Necesitamos centrar nuestras herramientas, técnicas y mentalidad en abordar la complejidad que siempre está en la raíz de nuestra disciplina.

El desarrollo de software como disciplina de ingeniería

Llegué a la conclusión de que esta suposición de que " **el desarrollo de software no es realmente ingeniería" es correcta en la práctica, pero muy errónea en principio** .

La forma en que trabajé durante la mayor parte de mi carrera ciertamente no se podía calificar de ingeniería, sin duda se acercaba más a una artesanía. Sin embargo, en la última parte de mi carrera comencé a pensar más conscientemente en cómo podía hacerlo mejor.

Comencé a adoptar un enfoque consciente y más racional para la toma de decisiones en todos los aspectos del desarrollo de software. Comencé a aplicar algunas heurísticas que me guiarían a mí y a los equipos con los que trabajaba de manera más confiable hacia mejores resultados. ¡Funciona!

Ingeniería de software moderna

En mi nuevo libro he intentado esbozar este enfoque organizado, pero pragmático y sencillo, del desarrollo de software, con el fin de capturar algunos principios que considero que son generales para todo el desarrollo de software, de manera que podamos adoptarlos y utilizarlos para orientarnos en la dirección de obtener resultados más exitosos.

Mi tesis es la siguiente: **si un enfoque de ingeniería para el desarrollo de software no nos ayuda a crear mejor software más rápido, entonces es incorrecto** y no califica como "Ingeniería".

Como la mayoría de los autores, estaba nervioso por cómo serían recibidas estas ideas, pero " Modern Software Engineering ", mi nuevo libro, está empezando a reunir excelentes críticas y la gente está encontrando las ideas que contiene tan útiles como yo. Si lo lees, espero que lo disfrutes.

Esta entrada fue publicada en Entrega continua , Integración continua , Cultura , Prácticas efectivas , Disciplina de ingeniería , Noticias personales , Diseño de software , Ingeniería de software , TDD y etiquetada BDD , Entrega continua , Desarrollo de software , Ingeniería de software , TDD . Guarda el enlace permanente .

12 respuestas a *¿Qué es la ingeniería de software moderna?*

AC *dice:*
2 de abril de 2022 a las 15:06

Hola,

Tú dices:

Entornos de bajo código que funcionan bajo el supuesto de que sabes exactamente lo que quieres al inicio de un proyecto y que ese requisito nunca cambiará: ¡buena suerte con esa idea!

¿Podrías explicarnos esto con más detalle? ¿Por qué crees que el low-code es una mala idea?