Pair Programming:
Poo & Pee adventures

@v4n3ss4ms

Vanessa - @v4n3ss4ms

Soluciono problemas de código e intento sobrevivir al apocalipsis.

¿Qué es eso del pair programming?

Según la Wikipedia

Es una técnica de desarrollo ágil en la que dos personas trabajan conjuntamente en un solo ordenador.

A la persona que escribe el código se le conoce como *driver* y la otra persona con el rol de *observer* es quien revisa lo qué se va escribiendo y aporta la estrategia.
Las personas intercambian los roles frecuentemente.

El mayor beneficio es que aunque se incrementa el coste persona-hora si se compara con el trabajo individual éste se ve compensado por el decremento de *defects*, lo cual a largo plazo aumenta el retorno de inversión.

Otros beneficios son, soluciones más diversas, satisfacción, aprendizaje, team building y mejoras en la comunicación.

Los roles

Roles: driver

Roles: driver

Es la persona que tiene el teclado, el que escribe:

- Está concentrada en resolver un problema concreto de código.
- Comparte la solución que va a tomar con su pair.
- No monopoliza el teclado.

Roles: navigator / observer

Roles: navigator / observer

Se encarga de:

- Controlar y reconducir la sesión.
- Aportar una visión global (pensamiento a largo plazo).
- Se asegura de mantener el foco en el problema en concreto.
- Evita los “*poyaques*”.
- No interrumpir al driver cuando está plenamente concentrado.
- Detectar si el driver está cansado o bloqueado.
- Evitar convertirse en un *eslint* humano.

En resumen

El pairing es una conversación entre dos desarrolladores que conjuntamente analizan, desarrollan y testean un “pedazo de código”.

Estas dos personas se comprometen a:

- Adoptar y rotar los roles.
- Tomar la iniciativa cuando su compañero se bloquea.
- Aportar ideas.
- Ayudar a mantener la concentración.

El bueno, el feo y el malo

El bueno, el feo y el malo

Lo bueno (el bueno)

- Se evitan los nichos de conocimiento (bus factor) y esto aumenta la resiliencia del equipo.
- Mejora la calidad del código, no dejas “para después” cosas que pueden acabar siendo un problema. Code reviews sobre la marcha.
- Mejora la moral del equipo sabiendo que será ayudado y ayudará.
- Onboarding más rápido.
- Descenso de bugs. Un estudio encontró que la reducción es de un 15%.
- Mantiene un WIP bajo.
- Facilita la concentración.

Los retos (el feo)

- Asumir que no hay una sola forma de abordar y ejecutar el pair programming.
- El pairing es agotador.
- Paciencia.
- Diferentes niveles en ciertas skills.
- No tienes tiempo para tus pensamientos.
- ¿Es que no soy capaz de hacer esto yo solo?
- Tienes que estar dispuesto a ser vulnerable.
- Evitar las dinámicas de poder.

Lo que hay que tener en cuenta (el malo)

- El pairing **NO** elimina los bugs.
- El pairing **NO** arregla una mala dirección de producto.
- El pairing cuesta dinero. El mismo estudio indica que el incremento de coste puede llegar al 100%. Muy superior en caso de una mala ejecución de pairing.
- El pairing **NO** convierte a los desarrolladores en senior por arte de magia.
- El pairing **NO** vale para todas las tareas.

Disclaimer

Disclaimer

Todos somos nuevos en algo, en el dominio o en la tecnología.

No usaré Senior / Junior.
No usaré “Persona que lleva más tiempo en el negocio” / “Recién llegado”.

Usaré...

La fuerza...

Tipos de pairing

Tipos de pairing

- Driver & Navigator: Pairing básico.
- Ping Pong: Técnica cercana al TDD.
- Strong-Style Pairing: La idea del navigator la escribe el driver. Jedi - Padawan.
- Tour guide: El navigator se convierte en el “guía turístico” del observer.
- Pair Development: Concepto explicado por Sarah Mei. Va más allá del puro desarrollo.
- Pair Debugging: No es un tipo de pairing al uso pero es algo a lo que todos recurrimos de vez en cuando.

Tipos de combinaciones

- Jedi & Jedi: Puede parecer la mejor de las opciones pero requiere que ambos Jedis asuman que pueden no saber algo (mostrarse vulnerables). Así mismo generalmente no emergen soluciones creativas ya que subyace un acuerdo no escrito de no cuestionarse mutuamente.
- Padawan & Padawan: No es lo más común pero es el ejemplo perfecto que demuestra que los resultados son mejores en pairing que por separado.
- Jedi & Padawan: Es la mejor de las opciones. Está fuertemente asociado al tipo de pairing Strong-Style. Puede generar situaciones de micro-management. Es fácil pasar del "*learning by doing*" al "*learning by watching*". Situación "*watch the master*".

Las dinámicas de poder

Formales

Las jerarquías formales derivan de la propia organización.

El ejemplo más habitual es el pairing de un manager con un *minion*…

Pair Programming: Poo & Pee adventures- @v4n3ss4ms

Informales

Estos son algunos ejemplos de jerarquías informales:

- Jedi - Padawan
- Hombre - no hombre
- Con titulación universitaria - sin titulación universitaria
- Persona blanca - persona no blanca

Dinámicas de poder

Las dinámicas de poder son interseccionales.
Cuando dos personas trabajan juntas suceden muchas de estas dinámicas e incluso se solapan. Ejemplos de situaciones:

- Dominar la sesión. Acaparar el teclado.
- Adoptar una posición y actitud de profesor de Master class.
- No escuchar y desestimar las sugerencias.

A veces es complicado vincular ciertos comportamientos a las jerarquías y uno tiende a pensar que simplemente “no te llevas bien con tu pareja”.
El problema subyacente está a menudo influenciado por un desequilibrio entre las dos personas.
Sarah Mei escribió un hilo sobre este tema en Twitter y también tiene una charla sobre dinámicas de poder en un entorno Agile.

Dinámicas de poder: ¿cómo conseguir el equilibrio?

El primer paso es que la persona que está en el lado con mayor peso de la balanza admita y reconozca su posición y privilegio derivado.
Sólo entonces podrá reflexionar sobre las interacciones con su pareja y cómo las dinámicas de poder afectan.

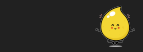
Reconocer la situación y adaptar el comportamiento para mejorar la colaboración no es fácil.
Es un camino duro de recorrer que requiere de mucho trabajo personal de reflexión.

A veces se puede recurrir a formaciones especializadas “anti-sesgos” que ayuden a crear equipos más cohesionados e inclusivos.

Dinámicas de poder: ¿estaré yo en el lado privilegiado?

Entre los materiales de un workshop anti sesgos hay un documento que nos ayuda a identificar nuestro poder y privilegio. Sólo hay que marcar lo que proceda:

Sources of privilege	Combined power and privilege
☐ Part of the dominant ethnic and/or racial group	☐ Educated
☐ Male	☐ Technically experienced
☐ Masculine	☐ Wealthy (compared to peers)
☐ Cisgender (your gender is the same as that assigned to you at birth)	☐ Management position
☐ Straight/heterosexual	☐ Professor, teacher, supervisor, teaching assistant, etc.
☐ Not disabled	☐ Head of family/household
☐ Neurotypical (society is designed for the way your brain works)	☐ Married
☐ A legal resident or citizen	☐ Widely recognized as an expert
☐ Speak the dominant language	☐ Large audience (social media following, fans, etc.)
☐ Speak with a high-status accent	☐ Access to media figures (reporters, TV shows, editors, etc.)
☐ Neither "too young" nor "too old"	☐ Connected to powerful people
☐ Certain height/size/shape	☐ Access to opportunities/networks
☐ Not a mother	☐ Any position in a hierarchy that is not the bottom of the hierarchy
☐ Not a caregiver	☐ _____
☐ From an upper or middle class family	☐ _____
☐ Dominant caste	



Dinámicas de poder: ¿estaré oprimiendo a algún colectivo?

También nos animan a no fomentar otro tipo de discriminación:

While you're trying to help one group, don't be:

- sexist
- homophobic
- transphobic
- racist
- ableist
- ageist
- classist
- casteist
- and don't criticize people's bodies or attractiveness

CC BY-SA Alan Levine, edited by FSC

Pairing, diversidad... ¿Qué?

En numerosos estudios se indica que la diversidad, inherente (raza, género, etc) o adquirida (experiencia, background cultural, etc) está asociada al éxito.

Hay un sesgo común que los psicólogos llaman heurística de la fluidez, que dice que preferimos la información que se procesa más fácilmente.
En un equipo homogéneo las personas se entienden fácilmente y la colaboración es fluida.
En un equipo heterogéneo se generan más dudas y debates. En definitiva, cuesta más entenderse.

No pain, no gain.

Cosas importantes que conviene no olvidar...

Cosas importantes que conviene no olvidar...

- Favorecer el aprendizaje: Poder preguntar sin tapujos. Ayudar a “perder el miedo” y generar confianza, por ejemplo podría subir código que aunque pudiera ser mejorable sigue siendo válido y “no rompe”.
- Diversidad del entorno de trabajo: No más guerras mi editor/IDE es mejor que el tuyo. El driver tiene que programar en lo que le resulte más cómodo.
- Efecto perrito de la pradera: Necesidad de asomar la cabeza y distraerse con otras cosas para coger perspectiva y frescura.
- Aprender a entender: Es necesario hacer el esfuerzo de entender el proceso mental de tu compañero. Así mismo tienes que ser capaz de expresar tu hilo de procesamiento.
- Generar un entorno seguro: Para que personas introvertidas, con ansiedad o neurodivergentes (autismo, ADHD...) puedan desarrollar su trabajo igual que el resto. Hay que asumir que el pairing es una herramienta fantástica sólo si nos sentimos cómodos.

El comodín del público

El comodín del público

- ¿Qué le aporta a un Jedi el pair programming?:
  - Romper nichos de conocimiento.
  - Reafirmar sus conocimientos.
  - Generar nuevas relaciones.
- ¿Todo el mundo vale para el pairing?: No. Hay que asumir que van a ser cuestionadas tus decisiones pasadas y actuales.
  Necesitas tener mucha paciencia.
  El pairing requiere de predisposición para el cambio, para mejorar, para recorrer un camino distinto al imaginado de la mano de otra persona y cuando uno es viejo los nuevos hábitos cuestan.

El comodín del público

- Dificultades:
  - Inseguridad.
  - Agotamiento.
  - Dificultad de encontrar tiempo y/o encaje con la otra persona.
  - Justificación económica.
- ¿Para qué es perfecto?: Onboardings.
- Preocupación: Muchos hombres dan por hecho que los no hombres sufren bastantes abusos de poder derivados de su condición de no hombres. Expresan su disconformidad y preocupación.
- ¿Qué es lo más importante?: ...

¿Qué es lo más importante?

¡La confianza!

Sin la confianza Pee y Poo están perdidos.

El pair programming es un marco y el contexto lo ponen las personas.

Enlaces de interés

- Pair Programming wikipedia. Programación en pareja wikipedia en español.
- Pair Programming: Pros, Cons, Best Practices
- When to pair program
- What is Pair Programming and How to Practice it in a Remote Team
- On Pair Programming
- Diverse Teams Feel Less Comfortable — and That's Why They Perform Better
- Pair Programming misconceptions
- Pair programming sucks
- Descubriendo el poder de Pair Programming
- My Experience with Pair Programming
- Effects of pair programming at the development team level: An experiment
- Impresiones sobre pair programming
- Pair programming: Mi guía práctica
- Llewellyn's strong-style pairing
- Ally Skills Workshop
- Ally Skills Workshop - Materiales
- The Benefits of Pair Debugging
- Maximizing the Gains and Minimizing the Pains of Diversity: A Policy Perspective
- Power dynamics in pairs and mobs
- Tim Ottinger's thoughts on Software Development

Más enlaces de interés

- Twitter - Sarah Mei: por qué Agile/XP falla en equipos heterogéneos
- Twitter - Sarah Mei: Common power dynamics in play in a pair programming situation
- Reddit - Why isn't pair programming common?
- PDF - The Costs and Benefits of Pair Programming
- PDF - Pair programming productivity: Novice–novice vs. expert–expert
- PDF - A five-stage model of the mental activities involved in directed skill acquisition

Imágenes

- Gifs - Giphy
- Vectores - vecteezy

GRACIAS

Mención especial a Noe, Henar, Alex e Iru por charlar conmigo, compartir sus experiencias, escucharme y estar ahí.

Muchas gracias a todas las personas que han contestado a la encuesta que lancé.

Muchas gracias a todes vosotres por estar aquí.

Pair Programming: Poo & Pee adventures- @v4n3ss4ms