Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223898475
RFC: ATI120618V12

Revista Dilemas Contemporáneos: Educación, Política y Valores.
http://www.dilemascontemporaneoseducacionpoliticayvalores.com/
Año: VIII Número: Edición Especial. Artículo no.:28 Período: Diciembre, 2020

TÍTULO: Programación colaborativa. De la necesidad de su uso a la psicología de sus interacciones.

AUTOR:

1. Dr. Ramón Ventura Roque Hernández.

RESUMEN: El desarrollo colaborativo se refiere al proceso de creación de software a través de la interacción simultánea de más de una persona; es así como los programadores colaboran entre ellos para obtener un sistema informático funcional y de calidad. Estos enfoques plantean serios desafíos. Entre ellos, se encuentran los aspectos humanos relacionados con las interacciones que los programadores establecen durante su trabajo. En este artículo se presentan dos formas de desarrollo colaborativo: la programación por pares y la programación mob. Posteriormente, se reflexiona sobre las ventajas y retos de cada una. Asimismo, se presentan cuatro maneras distintas de enfrentar sus desafíos: la etiqueta de colaboración, la programación sin ego, el engagement, y el trabajo en equipo.

PALABRAS CLAVES: Programación colaborativa, aspectos humanos del software, psicología de la programación, desarrollo de software, interacciones humanas.

TITLE: Collaborative programming. From the necessity of its use to the psychology of its interactions.

AUTHOR:

1. Dr. Ramón Ventura Roque Hernández.

ABSTRACT: Collaborative development refers to the process of creating software through the simultaneous interaction of more than one person; this is how the programmers collaborate among themselves to obtain a functional and quality computer system. These approaches pose serious challenges. Among them are the human aspects related to the interactions that programmers establish during their work. This article presents two forms of collaborative development: peer programming and mob programming. Later, they reflect on the advantages and challenges of each one. In addition, four different ways to meet your challenges are presented: the collaboration tag, egoless programming, engagement, and teamwork.

KEY WORDS: collaborative programming, human aspects of software, psychology of programming, software development, human interactions.

INTRODUCCIÓN.

El desarrollo colaborativo se refiere al proceso de creación de software a través de la interacción simultánea de más de una persona. En estos entornos de trabajo, los programadores colaboran para obtener sistemas informáticos funcionales y de calidad. No existe una manera única en que esta interacción se lleve a cabo. Tampoco existe una perspectiva única para estudiar estos fenómenos, pues el desarrollo colaborativo es un concepto amplio que se puede abordar desde diversos ángulos. En este trabajo se exploran algunas formas colaborativas de crear software, se reflexiona sobre los retos humanos que estas plantean y se presentan algunos elementos que ayudan a enfrentarlos desde el interior de los equipos.

DESARROLLO.

¿Programación individual o colectiva?

Tradicionalmente pensamos, que el desarrollo de software se puede realizar de manera individual y aislada de otras personas. Incluso, la formación profesional que recibimos en las aulas privilegia en

cierto sentido esta modalidad. Es verdad que existen tareas que se pueden realizar en solitario, especialmente cuando se trabaja en la creación de sistemas de software pequeños y sencillos; sin embargo, cuando el nivel de dificultad de los sistemas aumenta, ya sea por el número de requisitos que implementan, o bien, por la complejidad de sus funcionalidades, el software necesariamente se tiene que desarrollar en compañía de otras personas. De esta manera, la interacción humana en el proceso de creación de software es en muchas ocasiones, obligatoria.

¿Cuándo surgió la necesidad de trabajar colaborativamente?

Cuando iniciaron las primeras computadoras, hubo necesidad de crear programas para que estas funcionaran. En ese entonces, no existía la ingeniería de software como tal y la programación era un proceso más artesanal que metodológico. No fue hasta 1968, que la ingeniería del software surgió buscando ordenar las actividades relacionadas con la creación de programas de cómputo, orientándolas hacia la obtención de un producto de calidad, a través de un proceso repetible, que minimizará la probabilidad de fracaso. Su creación fue la respuesta a la llamada crisis del software, causada por sistemas complejos, de difícil implementación, que superan su presupuesto original, que no satisfacen los requerimientos para los que fueron creados y que demoran mucho en concluirse, o bien, no se concluyen nunca.

En la década de los ochenta y noventa, la complejidad de las aplicaciones informáticas creció todavía más y la importancia de la ingeniería de software fue evidente. Se hizo necesaria la colaboración de varias personas en proyectos de amplio alcance; sin embargo, no fue hasta el año 2001, en Utah, Estados Unidos, cuando se reconoció oficialmente que el software podía desarrollarse de mejores maneras. Se firmó el manifiesto ágil (Kent Beck et al., 2001), que significó un cambio de paradigma en la creación de programas informáticos. Se enfatizaron los aspectos humanos sobre los aspectos técnicos y se centró la atención en el software que funciona y agrega valor a quien lo usa, dejando así

de lado la documentación excesiva y los procedimientos densos y tardados. A esta filosofía se le denominó “agilidad”.

Privilegiando los aspectos humanos sobre los aspectos técnicos.

El manifiesto ágil valora en mayor medida aspectos como: los individuos y sus interacciones, las colaboraciones y la respuesta ante el cambio, sobre otros aspectos como: los procesos y las herramientas, las negociaciones contractuales y el seguimiento de planes inflexibles. Además, destaca elementos como la simplicidad, los individuos motivados, la buena comunicación cara a cara y la importancia de la reflexión y de la retroalimentación grupal. Con la firma de este manifiesto quedaba claro que el lado humano del software estaba siendo valorado y pronto, muchos enfoques de trabajo ágiles se apegarían en mayor o menor medida a esta filosofía.

La importancia de las interacciones humanas.

El desarrollo de software con frecuencia es un área incompatible con las necesidades de sus actores; por ejemplo, existen procesos fríos e inflexibles que no reconocen la faceta humana de los programadores; sin embargo, en un escenario de desarrollo de software, los actores a final de cuentas son personas trabajando para otras personas. Entonces no es entendible cómo son ellas mismas las que ignoran o soslayan las necesidades de los demás, o bien, se aprovechan ventajosamente de sus fortalezas.

En el proceso de creación de programas informáticos suelen existir presiones laborales, estrés, plazos urgentes, jornadas prolongadas e interacciones difíciles entre colegas. Estos elementos deben ponderarse en favor de los desarrolladores con el objetivo de lograr simultáneamente mayores niveles de productividad laboral y de bienestar personal.

Algunas formas colaborativas de creación de software.

Programación por pares.

La programación por pares es una técnica promovida por la metodología ágil denominada "Programación extrema", creada por Kent Beck (Kent; Beck & Andres, 2004; Zieris & Prechelt, 2019). Consiste en dos personas que crean software juntas compartiendo un mismo equipo de cómputo al mismo tiempo. En este enfoque existen dos roles: el conductor, que escribe el código, pues tiene el control del ratón y teclado; y el navegante, que guía el trabajo del conductor. Ambos roles se intercambian periódicamente. Aunque esta modalidad puede realizarse también de manera virtual, fue concebida para implementarse presencialmente.

Además de la programación por pares, la programación extrema recomienda otras prácticas con acentuación humana; por ejemplo, celebrar los logros con el resto del equipo de trabajo. Esto puede realizarse al final de cada una de las iteraciones, que son periodos cortos repetitivos en los que se va agregando funcionalidad y valor al software. Las celebraciones son actividades sencillas pero significativas que rompen la rutina de los colaboradores. Algunas ideas para celebrar son: compartir unas galletas, unos pastelitos, una comida, o hacer un brindis. Otra práctica humana recomendada por la Programación Extrema es evitar el estrés laboral a través del límite de la jornada laboral a 40 horas por semana cuando se trate de un trabajo de tiempo completo. Esto quiere decir, que las personas no deben trabajar horas extras fuera de su horario regular. En el caso extremo de que fuese absolutamente necesario hacerlo, se prohíben las horas extras por más de una semana.

Programación Mob.

La programación Mob (Pyhäjärvi & Falco, 2018; Zuill, 2014) se implementa cuando tres o más personas trabajan juntas en una misma computadora en el mismo espacio físico y desarrollando el mismo proyecto de programación con una filosofía común de unidad. El equipo de trabajo actúa como una sola persona: con una sola extensión telefónica y una misma cuenta de correo electrónico para

todos. En esta modalidad, también existe el rol de conductor, que es el encargado de usar teclado y ratón; un navegador que escucha, analiza y decide qué hacer; y una multitud o "mob", que observa, aporta, revisa, y da opiniones. En la programación mob, el entorno es un factor importante, pues se requieren instalaciones físicas especiales. Son necesarios, por ejemplo: un lugar privado e insonorizado para evitar interrupciones externas, una proyección ampliada para que todos puedan apreciar bien el código que se está escribiendo, un pizarrón o rotafolio de apoyo para el navegador y sillas suficientes para que se siente la multitud.

Ventajas de la buena colaboración.

La buena colaboración ha demostrado ventajas importantes; por ejemplo, entre los participantes mejora la comunicación, la productividad, las habilidades para aprender, el nivel de satisfacción, el conocimiento compartido, la incorporación de nuevos miembros al equipo, la calidad del software y el sentido de pertenencia. Por otra parte, disminuye el tiempo de desarrollo y las tasas de errores. En el caso concreto de la programación por pares, también se ha utilizado en la educación con muy buenos resultados (Hannay, Dybå, Arisholm, & Sjøberg, 2009); por ejemplo: ha logrado disminuir los índices de reprobación, ha contribuido favorablemente a las calificaciones de los estudiantes y ha sido un medio de inclusión de las mujeres a las ciencias de la computación (Ying et al., 2018).

Retos de los enfoques colaborativos.

Como toda interacción entre personas, la colaboración puede provocar situaciones sensibles debidas a las actitudes, a la conducta, a la personalidad, o al cansancio físico y mental de los participantes. Esto podría presentarse especialmente en la modalidad presencial de la programación por pares, ya que se basa en la convivencia de dos personas en un espacio reducido durante varias horas. Por otra parte, en los enfoques colaborativos es importante que todos los programadores se involucren activamente; es decir, que no adopten actitudes pasivas; no obstante, es común que algunas personas

participen poco, pierdan el interés o se distraigan (Buchan & Pearl, 2018). Esto ocurre con mayor frecuencia en la modalidad mob, en donde es fácil que una persona tome el control y el resto se subordine a ella.

Psicología de la creación colaborativa de software.

La psicología de la programación (Blackwell, Petre, & Church, 2019) es un área interdisciplinaria que se popularizó desde la década de los setenta. Se encarga de estudiar la cognición de los programadores de computadoras, así como herramientas y métodos para actividades relacionadas con la programación y su aprendizaje. Aborda también la usabilidad desde el punto de vista humano y una gran variedad de fenómenos, desde problemas de principiantes hasta el conocimiento de los expertos, desde la etapa de diseño hasta la etapa de pruebas y desde programas cortos e individuales hasta grandes sistemas creados en colaboración (Sajaniemi, 2008). Existen por lo menos cinco paradigmas psicológicos desde los cuales se abordan los problemas de la programación (Curtis, 1988):

1) Diferencias individuales.
2) Conducta grupal.
3) Conducta organizacional.
4) Factores humanos.
5) Ciencia cognitiva.

Para el caso que nos ocupa, la psicología de la programación ayuda a caracterizar, entender y enfrentar los desafíos que la programación colaborativa presenta a quienes la practican. A continuación, se discuten algunas formas para mejorar las interacciones en el entorno del desarrollo de software.

Espacio personal y etiqueta.

El espacio personal, o distancia interpersonal se refiere al área al alrededor de una persona, que contiene límites invisibles (Universitat de Barcelona, 2020). Al traspasarlos sin autorización, se produce en la persona una sensación de amenaza.

No se puede hablar de una distancia interpersonal de dimensiones iguales para todos. Esta distancia puede aumentar o disminuir en función de las situaciones y del tipo de interacciones que se tienen con otras personas. De manera particular, en la programación por pares, pueden existir invasiones al espacio personal con mayor frecuencia, dada la configuración de trabajo entre dos desarrolladores que están uno junto al otro utilizando una misma computadora. Kent Beck, creador de la Programación por pares, menciona que la cultura de cada participante debe ser tomada en cuenta para tener una interacción saludable; por ejemplo, menciona que en la cultura italiana, la cercanía es bien aceptada y favorece la comunicación. Lo contrario ocurre en la cultura danesa (Kent; Beck & Andres, 2004).

Para prevenir estas amenazas a la distancia interpersonal, existen algunas recomendaciones de etiqueta en la programación por pares (Rapid7, 2017); por ejemplo, hacer acuerdos sobre el espacio físico, expresar abiertamente cuando alguna situación cause incomodidad, y cuando exista algún desacuerdo con la otra persona, hablar siempre en términos de beneficio mutuo.

Entre las reglas de etiqueta para la programación por pares, también se recomienda reconocer a la pareja de trabajo con algún elogio o gesto agradable, y ser específico al referirse al código, mencionando siempre el nombre del archivo y el número de línea sobre el que se está hablando. Por otra parte, también se recomienda tener respeto y consideración por el compañero, así como acordar los tiempos de trabajo y descanso, para que ambos integrantes los respeten. Asimismo, es recomendable tener una buena higiene personal, no usar lociones con aroma fuerte, evitar comidas condimentadas con ajo o cebolla y utilizar pastillas para el aliento.

Egoless Programming.

Otra manera de apoyar las buenas interacciones es la programación sin ego o "egoless programming" (Weinberg, 1971), la cual es una filosofía que busca mejorar la interacción entre varios programadores que colaboran simultáneamente en proyectos de software. Esta filosofía busca reducir los comportamientos egoístas y ególatras que las personas solemos tener. Fue propuesta por Gerald M. Weinberg en la década de los setenta y consiste en diez principios o mandamientos cuyo objetivo es que los desarrolladores tengan normas comunes de pensamiento que hagan más llevadero su trato cotidiano:

1. Entiende y acepta que cometerás errores. - Gracias a nuestra naturaleza humana e imperfecta, siempre cometeremos algún error eventualmente.
2. Tú no eres tu código. - Es importante que los errores de software se descubran. Si alguien encuentra errores en el código de otra persona, es algo positivo para el proyecto. No es una situación personal ni es un insulto dirigido a alguien.
3. No importa cuánto karate sepas, siempre alguien sabrá más. - Todos aprendemos de todos. Si alguien tiene vastos conocimientos en el área de programación, seguramente se podrá aprender mucho de él.
4. No reescribas código sin consultarlo antes con alguien más. - Se debe tener la humildad de consultar a otra persona cualquier cambio, especialmente si se piensa que no es necesario.
5. Trata a los que saben menos que tú con respeto, educación y paciencia. - Es un principio básico para cultivar el buen trato y la tolerancia.
6. La única constante en el mundo es el cambio. - Lo único cierto es la incertidumbre y hay que aceptarla como algo que siempre estará presente en todo lo que se realice.
7. La única autoridad real deriva del conocimiento, no de la posición. - El conocimiento genera autoridad y respeto verdadero, incluso más que una posición importante de mando.

8. Lucha por lo que crees, pero acepta la derrota con deportividad.- No siempre las demás personas aceptarán lo que alguien sugiere. Sin embargo, estas situaciones no deben tomarse de manera personal. Tampoco es saludable adoptar actitudes de represalia.
9. No seas ese tipo aislado en la habitación. - Aunque se reconoce que la privacidad es importante, se recomienda buscar los momentos para interactuar y colaborar con otras personas.
10. Critica el código y no a la gente. - Los comentarios siempre deben ser positivos y deben girar en torno al código, buscando mejorarlo. Al decir algo sobre el código, siempre hay que relacionarlo con los estándares adoptados, con los requerimientos del programa o con el desempeño del programa.

Engagement.

El "Engagement" es un constructo psicológico que implica una conexión emocional con una actividad y hace referencia al nivel en el que un individuo se encuentra involucrado en ella. Para entender este constructo, se deben considerar componentes cognitivos, emocionales y conductuales. En el entorno de la creación colaborativa de software, el engagement implica compromiso y participación; por lo tanto, el equipo se beneficia cuando este aumenta entre sus integrantes. Para incrementarlo, se recomiendan actividades que promuevan la pertenencia al equipo, la conexión con el proyecto, las relaciones profundas entre los participantes, las interacciones de alto alcance y el reconocimiento personal y profesional.

Trabajo en equipo.

El trabajo en equipo se realiza por personas comprometidas con un objetivo común, que tienen capacidades complementarias y un sentido de responsabilidad compartida (Jaramillo Soloro, 2012). Para lograr un verdadero trabajo en equipo es importante construir confianza entre los participantes, establecer objetivos comunes, crear un sentido de pertenencia, involucrar a los miembros en las

decisiones, motivar la responsabilidad y el compromiso mutuo, impulsar la comunicación, aprovechar la diversidad, y celebrar los éxitos grupales (Gómez Pereira, 2020).

El trabajo en equipo es importante pues favorece la creatividad, el aprendizaje, la multiplicación de talentos y la eficacia en el trabajo. Puede decirse que se está trabajando efectivamente en equipo si se toman decisiones grupales, si las reuniones son productivas, si los participantes se escuchan entre sí y si se fomenta la creatividad y la innovación.

CONCLUSIONES.

La programación colaborativa es un enfoque que ha demostrado ser efectivo en la creación de sistemas informáticos de buena calidad en corto tiempo. Los equipos, que pueden trabajar bajo diversos enfoques organizativos, son más que la suma de sus colaboradores. Ahí radica el poder de esta forma de desarrollar software. Entonces, las interacciones entre desarrolladores son importantes y deben caracterizarse, entenderse y fortalecerse con el objetivo de lograr una convivencia saludable, así como mayores niveles de productividad.

La psicología de la programación nos ayuda en estas tareas. Los aspectos humanos relacionados con el software con frecuencia son minimizados; sin embargo, en las últimas dos décadas ha quedado evidenciada su valía y se prevé que en los próximos años crezca el interés por abordarlos formalmente en mayor medida de manera directa o indirecta.

REFERENCIAS BIBLIOGRÁFICAS.

1. Beck, Kent;, & Andres, C. (2004). *Extreme Programming Explained. Embrace change.* Estados Unidos: Addison Wesley.
2. Beck, Kent, Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifiesto ágil. Retrieved from Agilealliance.org website: https://agilemanifesto.org/iso/es/manifesto.html

3. Blackwell, A. F., Petre, M., & Church, L. (2019). Fifty years of the psychology of programming. *International Journal of Human-Computer Studies*, *131*(June), 52–63. https://doi.org/10.1016/j.ijhcs.2019.06.009
4. Buchan, J., & Pearl, M. (2018). Leveraging the mob mentality. *ACM International Conference Proceeding Series, Part F1377*. https://doi.org/10.1145/3210459.3210482
5. Curtis, B. (1988). Chapter 4 - Five Paradigms in the Psychology of Programming. In *Handbook of Human-Computer Interaction* (pp. 87–105).
6. Gómez Pereira, B. (2020). 10 claves del trabajo en equipo. Retrieved from Entrepreneur website: https://www.entrepreneur.com/article/267144
7. Hannay, J. E., Dybå, T., Arisholm, E., & Sjøberg, D. I. K. (2009). The effectiveness of pair programming: A meta-analysis. *Information and Software Technology*, *51*(7), 1110–1122. https://doi.org/https://doi.org/10.1016/j.infsof.2009.02.001
8. Jaramillo Soloro, R. M. (2012). Trabajo en Equipo. In *Subsecretaría de administración y finanzas, Dirección general de recursos humanos.*
9. Pyhäjärvi, M., & Falco, L. (2018). *Mob Programming Guidebook*.
10. Rapid7. (2017). Five Rules of Pair Programming Etiquette. Retrieved from https://blog.rapid7.com/2017/01/27/5-rules-of-pair-programming-etiquette/
11. Sajaniemi, J. (2008). Psychology of Programming: Looking into Programmers' Heads. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, *4*(1), 4–8. https://doi.org/10.17011/ht/urn.200804151349
12. Universitat de Barcelona. (2020). Psicología ambiental. Elementos básicos.
13. Weinberg, G. M. (1971). *Psychology of Programming*. Nueva York, Estados Unidos: John Wiley & Sons, Inc.

14. Ying, K. M., Pezzullo, L. G., Ahmed, M., Crompton, K., Blanchard, J., & Boyer, K. E. (2018). *In Their Own Words: Gender Differences in Student Perceptions of Pair Program-ming*. https://doi.org/10.1145/3287324.3287380
15. Zieris, F., & Prechelt, L. (2019). Does Pair Programming Pay Off? In *Rethinking Productivity in Software Engineering*.
16. Zuill, W. (2014). Mob Programming – A Whole Team Approach. In *Report*.

DATOS DEL AUTOR:

1. **Ramón Ventura Roque Hernández**. Doctor en Ciencias de la Computación y Doctor en Educación. Centro de trabajo: Universidad Autónoma de Tamaulipas. Actividad: Profesor Investigador. País: México. Correo Electrónico: rvhernandez@uat.edu.mx ramonroque@yahoo.com

RECIBIDO: 7 de noviembre del 2020. **APROBADO:** 27 de noviembre del 2020.