TRABAJO EN PAREJAS EN EQUIPOS SCRUM DE ORGANIZACIONES ÁGILES

Erich Bühler
Enterprise Agile Coach
erichbuhler@agilib.org

primera versión Mayo 2014, v1.4 2015

TRABAJO EN PARES Y LA CULTURA SCRUM 1

¿QUÉ ES EL TRABAJO EN PAREJAS? 2

PONIENDO LAS CREENCIAS A PRUEBA 3

COMIENZOS DEL TRABAJO EN PARES & SCRUM 4

RESULTADOS EN PRODUCTOS DE SOFTWARE 6

EFECTOS SOBRE EQUIPOS SCRUM 8

PRESIÓN SOCIAL DEL PAR 11

TRABAJO EN PAREJAS A DISTANCIA 14

NEGOCIACIÓN DE PARES 16

MODELO ECONÓMICO SUSTENTABLE 17

REFERENCIAS 20

Trabajo en Pares y la cultura Scrum

¿Te has preguntado a que se debe que el trabajo en parejas es parte la cultura de los equipos Ágiles?

Verás a continuación los motivos principales de porqué la implementación de características de software que brindan valor a un cliente y se lleven adelante en parejas es parte de una forma de trabajo efectiva y segura en equipos Scrum.

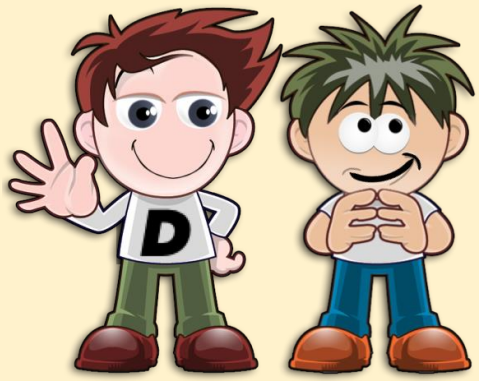
Pero antes, te recomiendo que leas mi libro digital sobre Modelos Sociales para Organizaciones Ágiles y Digital IT haciendo clic aquí

¿Qué es el trabajo en Parejas?

El trabajo en parejas es una **actividad social** en donde dos trabajadores del conocimiento trabajan codo a codo en **un solo computador**, colaborando continuamente en el diseño, código, interfaz gráfica, o cualquier otra actividad relacionada con el producto, para así concretar una característica que brinde valor a un cliente. En ella, se asume que la próxima tarea a desarrollar será tomada por dos personas con habilidades iguales o diferentes, sin que exista una selección arbitraria de las labores.

Trabajador de conocimiento: Cualquier persona que trabaje en un producto utilizando una herramienta de software y su conocimiento para implementar una característica que brinde valor a un cliente.

PONIENDO LAS CREENCIAS A PRUEBA



He decidido poner a prueba las prácticas de trabajo en pares, y para ello me he apoyado en 76 investigaciones que se han realizado en los últimos 25 años. Esto te servirá para conocer más sobre los pilares de la cultura Ágil.

Verás entre paréntesis el número de estudio, el que podrás encontrar al final del documento.

La mayor parte de las investigaciones que se analizan sobre trabajo en pares han destacado la **mejora en la calidad del producto, la moral de los equipos, distribución del conocimiento y amplia reducción del riesgo, así como el número total de defectos y aumento de valor entregado al cliente**.

Comienzos del trabajo en pares & Scrum

El trabajo en pares referente a productos de software nació de las técnicas **eXtreme Programming**, para posteriormente ser ampliamente adoptado por el marco de trabajo de **Scrum** (aunque éstas no pertenecen estrictamente al mismo).

Como te podrás imaginar, al inicio había bastantes dudas sobre la práctica ya que las personas creían que la cantidad del **esfuerzo** y la **coordinación** necesaria para realizar las tareas se vería incrementada debido a que se necesitarían **dos o más** personas para cumplir una única tarea.

El primer estudio realmente extenso que vimos fue realizado en 1999 por la **Universidad de Utah** [r01, r02, r03], donde se buscó aislar los costos y beneficios de ésta práctica. Para ello se pusieron personas a realizar tareas técnicas en pares y solas, con el fin de resolver diferentes pruebas de codificación.

	% de pruebas pasados por un solo estudiante	% de pruebas pasadas por un par
Grupo 1	73,4%	86,4%
Grupo 2	78,1%	88,6%
Grupo 3	70,4	87,1%
Grupo 4	78,1%	94,4%

Tabla 1: Porcentaje medio de pruebas pasadas por los estudiantes

Como puedes apreciar, estas indicaron que los pares incrementaron el rendimiento en al menos 15% cuando se comparaba con las personas solas. No obstante, el próximo apartado es un poco más interesante ya que analiza casos concretos de la industria del software y no experimentos en universidades.

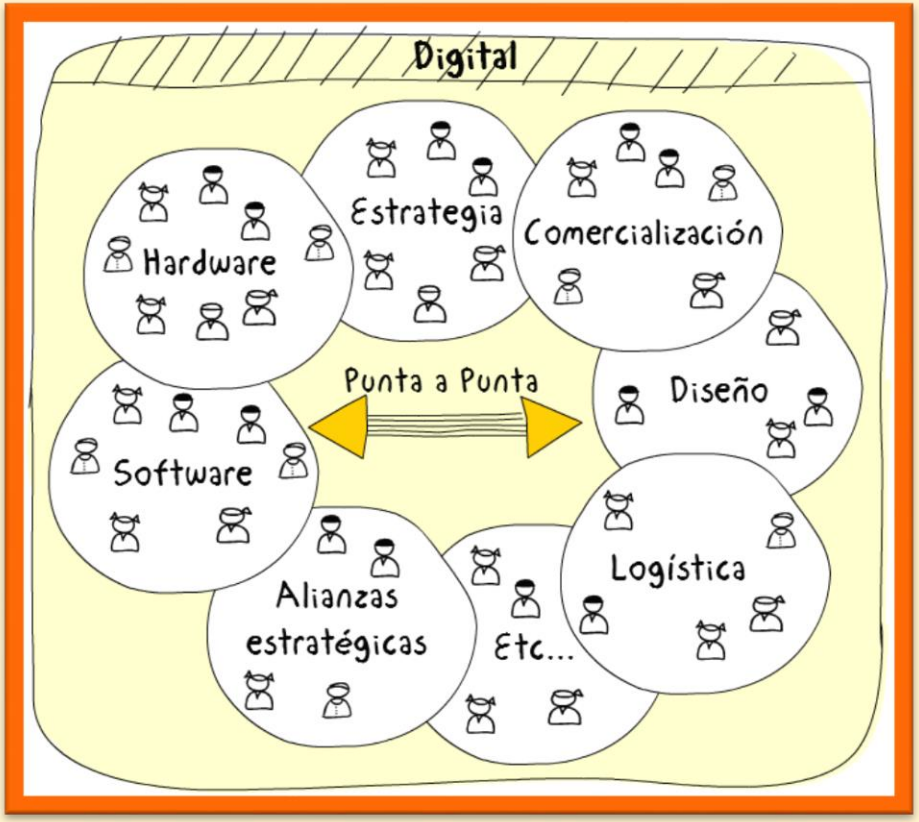
Pero antes de continuar, quiero contarte sobre otra **percepción** inicial que tienen las personas sobre las técnicas de pares. Muchas veces se cree que es útil en ciertos escenarios, por ejemplo, de provecho en tareas simples pero no en complejas [r04, r05, r06, r07]. Un ejemplo de **tarea compleja** es donde se requiere una habilidad muy específica que el resto del equipo desconoce.

Comienzos del trabajo en pares & Scrum

Esto pudo verificarse en un experimento con **295 consultores**, donde quedó claro lo contrario, que **existen mejoras sustanciales cuando se lleva adelante en cualquier tarea compleja** pero no así en las tareas muy simples [r08].

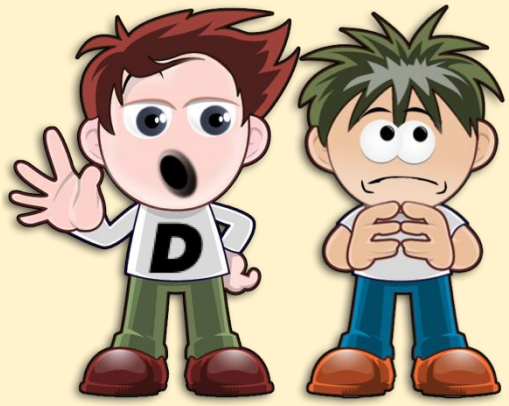
He observado en varios proyectos y empresas que las labores del día a día son particularmente complejas debido a su tamaño y variedad de elementos interconectados para resolver un problema específico, así como el aumento gradual de la presión que ejerce el mercado sobre la organización.

Es por ello que la utilización de las técnicas de pares son ampliamente **recomendadas para todos los miembros del equipo de la industria sin importar su área de conocimiento**.



RESULTADOS EN PRODUCTOS DE SOFTWARE

La **totalidad** de los equipos que han utilizado el trabajo en parejas en productos de software complejos han **mejorado sustancialmente la calidad del mismo** [r04, r09, r06, r07]. Uno de los tantos ejemplos es una empresa de **telecomunicaciones** de **Finlandia** cuyos empleados utilizaron **programación en parejas de forma exclusiva (24/7)** y debido a ello tuvieron **solamente 5 defectos en 1 año y medio** [r07].



Como te puedes imaginar, esto hizo que la compañía contase con mayores recursos y personas dedicadas a la **innovación** o a entregar mayor valor al cliente y menos empleados para arreglo de defectos.

Otro estudio en Finlandia [r05] demostró la misma cantidad de errores en comparación con trabajar en solitario pero **6 veces menos de defectos en un segundo proyecto**.

Efectos sobre equipos Scrum

Un efecto de la implementación de características en pares es que los equipos están **más motivados** de llevar adelante el "pairing" debido a que se **esparce conocimiento de forma continua** [r06] y sin interrupciones. Debido a esto se requiere que las **personas estén 100% dedicadas al equipo**, ya que de lo contrario, la **fluidez** de la **difusión** de conocimiento no es constante, produciendo así áreas "ciegas" y otros riesgos de bloqueos.

Es necesario indicar que el **proceso de selección** de nuevos empleados debe cambiar y ser **transparente**, dejando claro que la **unidad mínima** de trabajo es ahora el **par**. Esto debe ser así ya que de lo contrario puede haber **fricción** sobre las costumbres habituales de trabajo y la nueva forma que se le solicita al empleado.

Las vías de **pensamiento tradicional** y la educación actual se basan en modelos de éxito individuales, lo cual genera discordancia con el objetivo del trabajo en pares y las metas grupales.
Se puede encontrar reticencia también entre las personas existentes sobre adquirir ésta técnica, por lo que muy comúnmente se necesitarán cambios en los procesos y estándares de la organización, así como un apoyo activo de los líderes de la organización.

Efectos sobre equipos Scrum

Es importante destacar que en muchas empresas hay solamente **una persona en el equipo con conocimiento específico** (por ejemplo UX, Arquitectura, Seguridad, etc.), por lo que la consecuencia es un riesgo alto y un valor de **factor lotería** cercano a 1.

"***Factor lotería:*** *número de personas máximas que podrían desaparecer debido a recibir la lotería, sin impactar en la entrega o innovación del producto*"

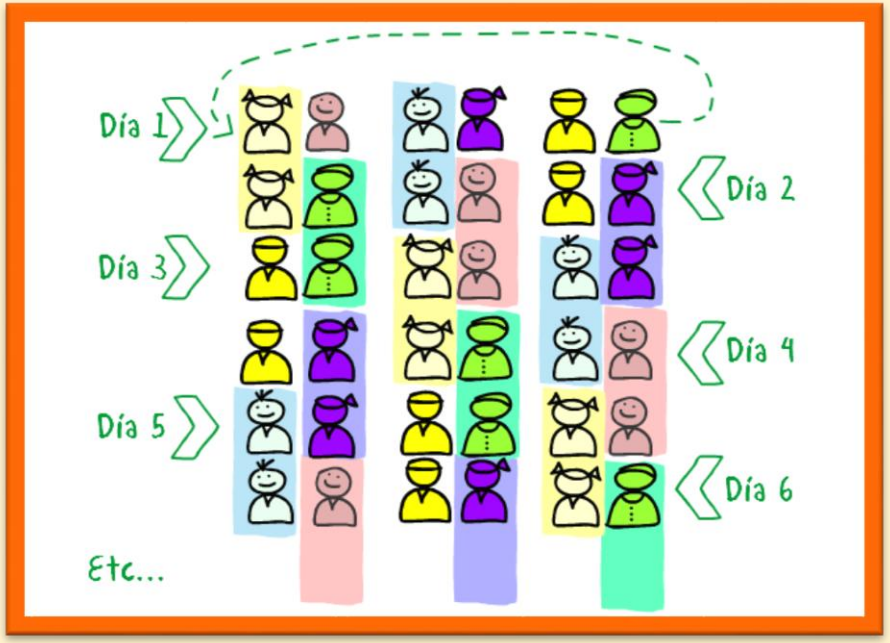
Teniendo en cuenta esto, verás que si hay más de un individuo con conocimiento específico, el **riesgo bajará exponencialmente**. Por el otro lado, he apreciado que tomar decisiones basados en **distintos puntos de vista** mejora la solución y aumenta el compromiso del equipo así como el el tomar posesión responsable de la solución elegida.

Los equipos con trabajo en pares tienden a **comprender mejor el código** en comparación con las laboras realizadas por una sola persona [r07, r10].

Esto es básicamente el resultado en sí de la forma de trabajo ya que ambos integrantes deberán **comprender** completamente la tarea y lógica antes de seguir adelante.

Efectos sobre equipos Scrum

Cuando se utiliza **el trabajo en parejas, los miembros sienten que han contribuido al progreso del equipo, lo que aumenta evidentemente la moral** [r07, r10] y disminuye la **paternidad de código**, cosa que hace que las personas sean mucho mas receptivas a modificar algo previamente realizado.



J. Vanhanen and H. Korpi han dejado clara la evidencia de que el uso del trabajo en pares **aumenta la disciplina en otras prácticas**, como ser pruebas primero (**TDD**), estándares de código, integración, etc. [r07, r10]. Estos últimos son **pilares fundamentales** que apoyan los equipos Scrum.

Como información adicional, ha habido un experimento menor que se ha llevado adelante por 8 trabajadores del conocimiento en Tailandia, el que demostró que la implementación de un producto llevó 4% más de tiempo al emplear pares, pero se obtuvieron 39% menos defectos en las **pruebas de aceptación** [r11].

PRESIÓN SOCIAL DEL PAR

Cuando dos o más personas trabajan en conjunto, éstas **se retro-alimentan de forma positiva** para generar presión social el uno sobre el otro. Se ha comprobado que los **trabajadores del conocimiento** llevan adelante las tareas más eficientemente y **obtienen resultados más creativos** cuando lo hacen en conjunto. Esto es en parte debido a que ellos no desean dejar a su pareja sola, lo cual aumenta el **foco**.

Un efecto colateral que se ha visto cuando se emplea la técnica es que las personas son **menos** propensas a las **interrupciones** (leer el correo electrónico, navegar por la web, Facebook, o hacer una llamada telefónicas). Ello refuerza siempre la **calidad** de los resultados obtenidos [r12].

Otro de los resultados [r12] nos demuestra que los trabajadores del conocimiento trabajan más intensamente y con mayor motivación durante la sesión que el trabajo en solitario.

La utilización de tiempos límite (**timebox**) también es recomendada para establecer plazos explícitos que motiven al par a que se focalice en la meta de concretar la tarea cuanto antes.

Presión social del par

Puedes a su vez investigar en internet sobre **prácticas** específicas de pares en equipos Scrum (**Mob programming**, etc.).

Déjame compartirte un estudio realizado por Alberto Sillitti y L. Plonka [r13] donde se vio que cuando un trabajador del conocimiento enfrenta un problema solo, éste ocupa una media de 33.3% de materialización en la solución (codificando, creando una interfaz de usuario, etc.) y 67.7% pensando sobre las diferentes opciones (buscando una alternativa en Internet, etc.).

Esto último es más evidente en trabajadores del conocimiento menos experimentados que aquellos con más experiencia.

Presión social del par

Tipo de trabajadores del conocimiento	Tiempo materializando la solución
Experto (solo)	33%
Novato (solo)	32%
Expertos (en pareja)	64%
Novatos (en pareja)	49%
Mixtos (en pareja)	75%

Tabla 2. Alberto Sillitti, 2012

Por su parte, cuando se trabaja en pares, el tiempo de codificación y el aprendizaje se ve incrementado a casi el doble así como un impacto en la calidad del producto [r13]. Otro ejemplo interesante es que los trabajadores del conocimiento novatos cuando están en parejas obtienen niveles de calidad similar a lo de los expertos.

TRABAJO EN PAREJAS A DISTANCIA

He visto a varios equipos de organizaciones que cuentan con personas que están en otros sitios geográficos. Aunque esto puede no ser tu caso, sé que es común el necesitar trabajar con individuos que se encuentren temporalmente o de forma indefinida a varios kilómetros de distancia.

Para el primer caso, siempre defino una **regla única** que consiste en que cuando un equipo trabaje en pares, uno de sus integrantes deberá estar siempre de forma local.

"Un par deberá estar siempre local para actualizar así los tableros visuales de papel"

Esto se hace así para que exista una persona encargada de actualizar los tableros visuales en papel, lo que se ha demostrado ampliamente que **disminuye la complejidad**.

Finalmente, el especialista Nick Flor nos ha dejado clara la importancia del trabajo en pares en los equipos distribuidos ya que ello apoya el espacio visual, auditivo y manual perdido [r14].

Estos canales permiten a los pares prestar colaboración y utilizar **expresiones sensoriales** sutiles para así apoyar el **intercambio de información**. Por ejemplo, algunos **gestos sutiles** como un movimiento de cabeza

o un murmullo podrían activar un **intercambio de información** en la pareja. Una sugerencia es la utilización de imágenes transparentes del otro miembro del par sobre la pantalla durante la sesión donde se trabaja, ya que ha mostrado ser un apoyo para así amplificar la **sinergia** entre ambas personas [r15].

Hay más estudios que han recalcado que el trabajo en parejas en un producto de software en entornos distribuidos hace la **integración del código** mucho más sencilla en comparación con lo producido por un trabajador único.

Esto puede ser de vital importancia en compañías que comienzan su travesía hacia la agilidad, donde la integración de productos es comúnmente compleja.

NEGOCIACIÓN DE PARES

¿Has escuchado el término de Negociación de pares? Ella es una expresión que se utiliza para describir la forma en que **dos** trabajadores del conocimiento llegan a una **mejor solución** si lo hacen juntos en vez de cada uno por su parte o conectando al principio y/o luego al final.

Cuando ellos trabajan de forma fluida, cada persona aporta su propio conjunto de **habilidades, capacidades y perspectivas** ya que comparten la misma meta para completar la tarea. Cada individuo tiene su forma de abordar el tema, por lo que ambos deberán siempre **negociar** con el fin de encontrar la mejor solución.

En la negociación se evalúan las **alternativas**, en comparación a cuando se trabaja solo, donde se tiende a elegir la **primera aproximación** que viene a la mente.

Modelo económico sustentable

He visto varios estudios que muestran que la **programación en pares** puede tomar **más esfuerzo** pero **produce un producto de mayor calidad.**

Debido a que hay **menos interrupciones** y dos cabezas pensando en como enfrentar la solución, los tiempos necesarios para completar la misma se ven disminuidos a casi la mitad.

El **equilibrio entre el aumento de velocidad y calidad** frente a un aumento de las personas necesarias es **una pregunta recurrente** y que ha sido examinada en dos **modelos económicos** diferentes [r16, r17].

El modelo desarrollado por **Erdogmus** [r16] es el primero y está basado en el **valor presente neto** para examinar la viabilidad económica del trabajo en parejas en productos de software.

Éste se basa en que el **valor reside en la capacidad de un par de entregar la característica requerida y percibir su correspondiente recompensa** (incluso aunque las entregas no contengan la totalidad de las características). Aquí se ha hecho un supuesto que reside en que cada defecto que se le escape a la organización le costará 33 horas-hombre [r18] y que a los pares les llevará 15% más de horas-hombre el completar la tarea [r02] que con un solo trabajadores del conocimiento.

Modelo económico sustentable

El resultado final en este caso ha sido que **el trabajo en pares es una forma viable alternativa a su contrapar de 1 persona.**

Padberg y Müller [r17] han creado un modelo similar en valor presente neto; estos han agregado como elemento suplementario la **presión que ejerce el mercado** en el desarrollo de un producto.

Aquí a diferencia se ha tomado la hipótesis que cada defecto que se le escape a la compañía le costará de 5 a 20 horas hombre [r18, r19].

Un término interesante que podrás emplear y que emana del artículo es lo que llaman la **ventaja de la velocidad del par**.

Finalmente, un tercer estudio ha sido el de Nosek [r20] que ha empleado valores más conservadores pero agresivos que los expresados por Williams [r02] y apoya la misma conclusión:

"Cuando la presión del mercado es intensa, agregar pares al desarrollo de un producto puede acelerar la entrega de valor a costa de un incremento en el número de las personas requeridas".

¡Espero que el trabajo en parejas apoye el éxito de tu organización!

Erich Bühler es director de Ágil Ibérica (***www.Agilib.org***) y *Enterprise Agile Coach* con más de 20 años de experiencia en la industria. Publicó en 2002 el primer libro en español sobre .NET, ha trabajado como profesor Universitario y consultor independiente para diferentes tipos de organizaciones en Inglaterra, España, Malta, Uruguay y Chile, y se especializa en apoyar a las compañías en los diferentes pasos que deseen dar en su camino hacia la agilidad y el éxito.

Visite su blog: BLOG

Puede contactarlo en erichbuhler@agilib.org

+34.609.369.263

Referencias

Las referencias que comienzan por [r..] han sido mencionados en el artículo, mientras que los restantes han servido de apoyo para el mismo.

[r01]E. Arisholm, H. Gallis, T. Dybå, and D. Sjøberg, "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," IEEE Transacations in Software Engineering, vol. 33, no. 2, pp. 65-86, February 2007. [2] P. Baheti, E. Gehringer, and D. Stotts, "Exploring the Efficacy of Distributed Pair Programming," in Extreme Programming/Agile Universe, Chicago, IL, 2002, pp. 208- 220. [3] P. Baheti, L. Williams, E. Gehringer, and D. Stotts, "Exploring Pair Programming in Distributed Object-Oriented Team Projects," in OOPSLA Educator's Syposium, Seattle, WA, 2002. [4] K. Beck, Extreme Programming Explained: Embrace Change, Second ed. Reading, MA: Addison-Wesley, 2005. [5] K. Beck, Extreme Programming Explained: Embrace Change. Reading, MA: AddisonWesley, 2000. [6] A. Belshee, "Promiscuous pairing and beginner's mind: embrace inexperience," in Agile Conference 2005, Denver, CO, 2005, pp. 125 - 131 [7] S. B. Berenson, L. Williams, and K. M. Slaten, "Using Pair Programming and Agile Development Methods in a University Software Engineering Course to Develop a Model of Social Interactions," in Crossing Cultures, Changing Lives Conference, Oxford, UK, 2005, p. to appear. [8] F. P. Brooks, Mythical Man Month: Essays on Software Engineering. Reading, MA: Addison-Wesley, 1975. [9] J. Carver, L. Henderson, L. He, J. Hodges, and D. Reese, "Increased Retention of Early Computer Science and Software Engineering Students Using Pair Programming," in Conference on Software Engineering Education and Training, Dublin, Ireland, 2007, pp. 115 - 122 [10] J. Chao and G. Atli, " Critical personality traits in successful pair programming," in Agile 2006, Minneapolis, MN, 2006, p. electronic proceedings. [11] J. Chong and T. Hurlbutt, "The Social Dynamics of Pair Programming," in International Conference on Software Engineering (ICSE) 2007, Minneapolis, MN, 2007, pp. 354-363 [r12] J. Chong and R. Siino, "Interruptions on Software Teams: A Comparison of Paired and Solo Programmers," in Computer Supported Collaborative Work (CSCW) 2006, Banff, Alberta, Canada, 2006, pp. 29-38. [13] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," in Extreme Programming and Flexible Processes in Software Engineering (XP2000), Cagliari, Sardinia, Italy, 2000. [14] L. L. Constantine, Constantine on Peopleware. Englewood Cliffs, NJ: Yourdon Press, 1995. [15] J. O. Coplien, "A Development Process Generative Pattern Language," in Pattern Languages of Program Design, James O. Coplien and Douglas C. Schmidt, Ed. Reading, MA: Addison-Wesley, 1995, pp. 183-237. [16] J. O. Coplien and N. B. Harrison, Organizational Patterns of Agile Software Development Upper Saddle River, NJ: Pearson Prentice Hall, 2005. [r04] T. Dybå, E. Arisholm, D. Sjøberg, J. Hannay, and F. Shull, "Are Two Heads Better Than One? On the Effectiveness of Pair Programming," IEEE Software, vol. 24, no. 6, pp. 12- 15, November/December 2007. [r16] H. Erdogmus and L. Williams, "The Economics of Software Development by Pair Programmers," The Engineering Economist, vol. 48, no. 4, pp. 283-319, 2003. [19] M. E. Fagan, "Advances in Software Inspection," IEEE Transactions on Software Engineering, vol. 12, no. 7, pp. 744-751, July 1986. [r14] N. Flor, "Globally Distributed Software Development and Pair Programming," Communications of the ACM, vol. 49, no. 10, pp. [r07] 57-58, October 2006. [21] S. Freudenberg, P. Romero, and B. du Boulay, "'Talking the talk': Is Intermediate-level conversation the key to the pair programming success story?," in Agile 2007, Washington, DC, 2007, pp. 84-91. [22] T. Frever and P. Ingalls, "The pairing session as the atomic unit of work," in Agile Conference, Minneapolis, 2006, p. electronic proceedings. [23] B. Hanks, "Problems Encountered by Novice Pair Programmers," in International Computing Education Research Workshop, Atlanta, GA, 2007, pp. 159 - 164. [24] B. Hanks, "Student Performance in CS1 with Distributed Pair Programming," in Innovation and Technology in Computer Science Education (ITiCSE) 2005, Monte de Caparica, Portugal, 2005, pp. 316-320. [25] C.-w. Ho, S. Raha, E. Gehringer, and L. Williams, "Sangam: A Distributed Pair Programming Plug-in for Eclipse," in Eclipse Technology Exchange at Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) 2004., Vancouver, BC, 2004. [26] A. Höfer, "Video Analysis of Pair Programming," in Workshop on Scrutinizing Agile Practices at the International Conference on Software Engineering, Leipzig, Germany, 2008, pp. 37-41. [r05] H. Hulkko and P. Abrahamsson, "A Multiple Case Study on the Impact of Pair Programming on Product Quality," in International Conference on Software Engineering (ICSE) 2005, St. Louis, Missouri, USA, 2005, pp. 495-504. [r18] W. S. Humphrey, A Discipline for Software Engineering. Reading, MA: Addison Wesley, 1995. [r19] W. S. Humphrey, Managing the Software Process. Reading,

Massachusetts: AddisonWesley, 1989. [[r09]] R. Jensen, "A Pair Programming Experience," in Crosstalk, March 2003. [31] A. Joseph and M. Payne, "Group Dynamics and Collaborative Group Performance," in Thirty-fourth SIGCSE Technical Symposium on Computer Science Education, Reno, NV, March 2003, pp. 368-371. [32] B. W. Kernighan and R. Pike, The Practice of Programming. Reading, Massachusetts: Addison-Wesley, 1999. [33] M. Lacey, "Adventures in Promiscuous Pairing: Seeking Beginner's Mind," in Agile 2006, Minneapolis, 2006, pp. 263 - 269. [34] L. Layman, "Changing Students' Perceptions: An Analysis of the Supplementary Benefits of Collaborative Software Development," in 19th Conference on Software Engineering Education and Training (CSEE&T '06), Turtle Bay, Hawaii, 2006, pp. 156- 166. [35] L. Layman, L. Williams, J. Osborne, S. Berenson, K. Slaten, and M. Vouk, "How and Why Collaborative Software Development Impacts the Software Engineering Course," in Frontiers in Education, Indianapolis, IN, 2005, pp. T4C 9-14. [r06] G. Luck, "Subclassing XP: breaking its rules the right way," in Agile Development Conference, Salt Lake City, UT, 2004, pp. 114 - 119 [37] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Effect of Pair Programming on Performance in an Introductory Programming Course," in ACM Special Interest Group of Computer Science Educators, Covington, KY, 2002, pp. 38-42. [38] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Impact of Pair Programming on Student Performance, Perception and Persistence," in International Conference on Software Engineering 2003, Portland, Oregon, 2003, pp. 602 - 607 [39] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "Pair Programming Improves Stuent Retention, Confidence, and Program Quality," Communications of the ACM, vol. 49, no. 8, pp. 90-95, August 2006. [40] E. Mendes, L. Al-Fakhri, and A. Luxton-Reilly, "A Replicated Experiment of Pair Programming in a 2nd year Software Development and Design Computer Science Course," in Innovation and Technology in Computer Science Education (ITiCSE) 2006, Bologna, Italy, 2006, pp. 108-112. [41] M. Mujeeb-u-Rehman, X. Yang, J. Dong, and M. Abdul Ghafoor, "Heterogeneous and homogenous pairs in pair programming: an empirical analysis," in Canadian Conference on Electrical and Computer Engineering 2005, Saskatchewan, Canada, 2005, pp. 1116 - 1119 [42] N. Nagappan, L. Williams, M. Ferzli, K. Yang, E. Wiebe, C. Miller, and S. Balik, "Improving the CS1 Experience with Pair Programming," in ACM Special Interest Group Computer Science Education (SIGCSE) 2003, Reno, 2003, pp. 359 - 362. [43] H. Natsu, J. Favela, A. Morán, D. Decouchant, and A. Martinez-Enriquez, "Distributed Pair Programming on the Web," in Mexican International Conference on Computer Science (ENC) 2003, Ciencias de la Computacion, CICESE, Mexico, 2003, pp. 81-88. [r15] K. Navoraphan, E. F. Gehringer, J. Culp, K. Gyllstrom, and D. Stotts, "Next-generation DPP with Sangam and Facetop," in OOPSLA workshop on eclipse technology eXchange, Portland, Oregon, 2006, pp. 6-10. [45] J. Nawrocki and A. Wojciechowski, "Experimental Evaluation of Pair Programming," in European Software Control and Metrics (ESCOM 2001), London, England, 2001. [r20] J. T. Nosek, "The Case for Collaborative Programming," Communications of the ACM, vol. 41, no. 3, pp. 105-108, 1998. [47] D. Oblinger, "Boomers, Gen-Xers, and Millennials: Understanding the New Students," Educause Review, vol. 38, no. 4, pp. 37-47, July/August 2003. [r17] F. Padberg and M. Müller, "Analyzing the Cost and Benefit of Pair Programming," in International Software Metrics Symposium (METRICS) 2003, Sydney, Australia, 2003, pp. 166 – 177, [49] A. Pandey, C. Miklos, M. Paul, N. Kameli, F. Boudigou, V. Vijay, A. Eapen, I. Sutedjo, and W. Mcdermott, "Application of tightly coupled engineering team for development of test automation software - a real world experience," in Computer Software and Applications Conference (COMPSAC) 2003, Dallas, TX, 2003, pp. 56 - 63 [r11] M. Phongpaibul and B. Boehm, "An Empirical Comparison Between Pair Development and Software Inspection in Thailand," in International Symposium on Empirical Software Engineering, Rio de Janeiro, 2006, pp. 85-94. [51] K. Schwaber and M. Beedle, Agile Software Development with SCRUM. Upper Saddle River, NJ: Prentice-Hall, 2002. [52] B. Simon and B. Hanks, "First Year Students' Impressions of Pair Programming in CS1," in International Computing Education Research Workshop, Atlanta, GA, 2007, pp. 73- 86. [53] K. M. Slaten, M. Droujkova, S. Berenson, L. Williams, and L. Layman, "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction," in Agile 2005, Denver, CO, 2005, pp. 323-330. [54] H. Srikanth, L. Williams, E. Wiebe, C. Miller, and S. Balik, "On Pair Rotation in the Computer Science Course," in Conference on Software Engineering Education and Training, Norfolk, VA, 2004, pp. 144 - 149. [55] G. Succi, M. Marchesi, W. Pedrycz, and L. Williams, "Preliminary analysis of the effects of pair programming on job satisfaction," in Fourth International Conference on eXtreme Programming and Agile Processes in Software engineering (XP2002), Sardinia, Italy, 2002, pp. 212-215. [56] T. Van Toll III, R. Lee, and T. Ahlswede, "Evaluating the Usefulness of Pair Programming in a Classroom Setting," in International Conference on Computer and Information Science (ICIS) 2007, Melbourne, Qld. , 2007, pp. 302 - 308 [57] J. Vanhanen and H. Korpi, "Experiences of Using Pair Programming in an Agile Project," in 40th Annual Hawaii International Conference on System Sciences (HICSS) 2007 Hawaii, 2007, pp. 274b - 274b [r10] J. Vanhanen and C. Lassenius, "Perceived Effects of Pair Programming in an Industrial Context," in 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, 2007. , Lubeck 2007, pp. 211 - 218 [59] J. Vanhanen, C. Lassenius, and M. Mäntylä, "Issues and Tactics when Adopting Pair Programming: A Longitudinal Case Study," in International Conference on Software Engineering Advances (ICSEA) 2007, Cap Esterel, French Riviera, France, 2007, p. 70. [60] G. M. Weinberg, The Psychology of Computer Programming Silver Anniversary Edition. New York: Dorset House Publishing, 1998. [61] L. Williams and R. Kessler, Pair Programming Illuminated. Reading, Massachusetts: Addison Wesley, 2003. [r01] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair-Programming," IEEE Software, vol. 17, no. 4, pp. 19-25, July/August 2000 2000. [63] L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamsson, "Toward a Framework for Evaluating Extreme Programming," in Empirical Assessment in Software Eng. (EASE) 2004, Edinburgh, Scot., 2004, pp. 11-20. [64] L. Williams, L. Layman, J. Osborne, and N. Katira, "Examining the Compatibility of Student Pair Programmers," in Agile 2006, Minneapolis, MN, 2006, pp. 411-420. [65] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner, "Building PairProgramming Knowledge Through a Family of Experiments," in International Symposium on Empirical Software Engineering (ISESE) 2003, Rome, Italy, 2003, pp. 143-152. [66] L. Williams, A. Shukla, and A. Antón, "An Initial Exploration of the Relationship Between Pair Programming and Brook's Law," in Agile Development Conference 2004, Salt Lake City, 2004, pp. 11-20. [67] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller, "In Support of Pair Programming in the Introductory Computer Science Course," Computer Science Education, vol. 12, no. 3, pp. 197-212, 2002. [r02] L. A. Williams, "The Collaborative Software Process," in Department of Computer Science Salt Lake City, UT: University of Utah, 2000. [r03] L. A. Williams and R. R. Kessler, "All I Ever Needed to Know About Pair Programming I Learned in

Kindergarten," in Communications of the ACM. vol. 43, 2000, pp. 108-114. [70] S. Xu and V. Rajlich, "Pair Programming in Graduate Software Engineering Course Projects," in Frontiers in Education, Indianapolis, 2005, pp. F1G-7-F1G12, [r12]Alberto Sillitti y L. Plonka (2011)