

Pensamientos de un tester ágil pragmático

El libro del blog "Tengo una idea"

Germán Braun

Pensamientos de un tester ágil pragmático

El libro del blog "Tengo una idea"

Germán Braun

Este libro está a la venta en <http://leanpub.com/pensamientosdeuntesterquequieresergile>

Esta versión se publicó en 2014-01-23



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](#)

¡Twitea sobre el libro!

Por favor ayuda a Germán Braun hablando sobre el libro en [Twitter](#)!

El hashtag sugerido para este libro es [#BlogTengaunaidea](#).

Descubre lo que otra gente está diciendo sobre el libro haciendo click en este enlace para buscar el hashtag en Twitter:

<https://twitter.com/search?q=#BlogTengaunaidea>

Índice general

Tengo una idea	1
Mis razones para escribir	1
Responsabilidad, cambios y motivación	1
Talent Pool	2
Tengo un problema	4
Tengo un problema (remasterizado)	4
Equivocarse	4
No escribirás con errores ortográficos	5
Stop Starting, Start Finishing	6
El condensador de flujo para nuestros proyectos	7
No revisaré mails al inicio del día	8
 Testing	 9
Hacer testing es también capturar conocimiento	9
Testing Agile: conceptos y principios	9
Hacer testing es también capturar conocimiento, otra vuelta	10
Basta de “Dev vs QA”	12
Principios escenciales para Automatización de Pruebas Funcionales	13
¿Por qué los testers necesitan aprender continuamente?	13
Never lose a BUG again	14
What's a Tester without a QA Team? (hacia un entorno ágil)	14
Habilidades de un tester	15
Tester World-Class	17
La T de Testing	17
Mi responsabilidad como tester	19
Una experiencia aplicando Kanban en Testing	19
Dar feedback continuo	22
Mind Mapping en Testing	23
Testing + Pensamiento Lateral = “Testing Lateral”	25
Hasta dónde hemos llegado	27
 Agilidad	 29
Agile is in the air	29

ÍNDICE GENERAL

Curso CSM	29
Kanban Maturity Chart	33
Mapas de Impacto	34
We don't know exactly where we'll go but small steps ensure we do not fall down	36
Así comienza el camino...	37
¿Para qué Scrum?	39
¡Necesitamos tener un tablero y una metodología, como sea!	39
Entrevistas y Aportes	40
Entrada en el blog de PetroVR	40
Testing: 6 preguntas a Ernesto Kiszkurno de Pragma Consultores	41
Desarrollar Software: 6 preguntas a Leandro Caniglia de Caesar Systems	42
Agile: 6 preguntas a Thomas Wallet de Pragma Consultores	44
Creatividad y Pensamiento	46
How do I find the next idea?	46
Mantenerse creativo	46
Mantenerse creativo (2)	46
Mantenerse creativo (3)	48
Mantenerse creativo (4)	49

Tengo una idea

Como el título lo indica, esta capítulo abarca entradas del blog que tratan sobre temas relacionados a responsabilidades, equipos, formas de trabajo e ideas en general.

Mis razones para escribir

¿Por qué estoy escribiendo?, ¿cuáles son las razones? Entonces identifiqué las siguientes (más representativas):

1. **Investigar:** me gusta hacerlo y el blog ayuda a plasmarlo. La gente que trabaja en Informática debe investigar todo el tiempo. El constante avance en materia de Ciencia y Tecnología requiere que todo profesional esté actualizado para intentar llegar a la “cresta de la ola”.
2. **Aprender:** El blog nos permite aprender sobre los temas que nos interesan, gestionarlo, escribirlo y compartirlo.
3. **Hacer Comunidad:** contactar gente con la que tengamos temas en común, que quiera compartir su conocimiento y que haga comunidad.

Cualquier motivación es válida para escribir. Sin embargo, creo que lo más importante y, lo que me motiva a seguir haciéndolo, es el feedback que se obtiene de los lectores (cualquiera sea la cantidad)

Responsabilidad, cambios y motivación

Buscando videos en [Youtube¹](#), me encontré con un [keynote de Alan Cyment en el Scrum Gathering 2012²](#). Lo comencé a ver y la verdad que me enganché.

Cyment comienza dando un enfoque interesante sobre Scrum, cuál es su definición y su espíritu. La charla continua, y luego de contar algunas otras experiencias, hablar sobre las culturas organizacionales, Scrum fingido y otras yerbas, introduce un tema que me pareció muy importante y que me inquieta hace rato. Tiene que ver con el rol de las personas en las organizaciones y puntualmente, en los equipo de trabajo. Aquí es donde me quiero detener ya que, además, coincido bastante en lo que Cyment dice, más allá de alguna que otra postura extrema o solución propuesta. Algo al respecto ya había mencionado [aquí³](#).

Lo resumo en las siguientes tres bullets:

¹<http://www.youtube.com/>

²<http://www.youtube.com/watch?v=MPc79nlL2Xw>

³<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

1. Debería ser **responsable de mi laburo**. Un tema que aquí se presenta es: “el problema siempre lo tiene el otro” y, si algo tiene que cambiar, ese no soy yo.
2. Debería poder **gestionar mis cambios**. Si algo no me gusta, trato de de cambiarlo introduciendo mejoras parciales y continuas o tratando de dar un cambio brusco para que la cosa cambie radicalmente.
3. Debería poder identificar qué me **motiva**. Cada uno puede tener motivaciones distintas por las cuales está en ese trabajo/empresa/equipo. En general, son casi las mismas (ambiente laboral, estabilidad, dinero, etc) aunque pueden estar en distinto orden de consideración.

Por último, y a modo de apreciación personal, creo que cada uno de nosotros comienza a crecer profesionalmente cuando se hace cargo y responsable de su propia carrera.

Talent Pool

Hace unos días, en mi trabajo, estuve presenciando una charla interesante sobre [NoSQL⁴](#). El orador era [Cristobal Pedregal Martín⁵](#). Sinceramente, no conocía mucho sobre el tema y, por lo tanto, no voy a entrar mucho en detalle sobre el concepto en este post. Más allá del tema en cuestión, estas exposiciones son oportunidades para abrirse a nuevos conocimientos y relacionarlos con otros ya existentes. Personalmente, en estas charlas me fijo mucho en cómo está estructurada la presentación, qué ejemplos se usan para explicar los conceptos y cómo el orador se dirige a la audiencia. Cómo hacer una buena presentación no es un tema que esté en condiciones de abordar profundamente, sin embargo, creo que hay que entrenarse al respecto para saber hablar delante de una audiencia.

⁴<http://en.wikipedia.org/wiki/NoSQL>

⁵<http://www.cs.unm.edu/~cris/>



Volviendo a la charla, y a lo que motivó esta entrada, quería mencionar lo que [Pedragal Martín⁶](#) nombró como _Talent Pool. _El concepto es relativamente simple. Sin embargo, me interesó el contexto en el cuál lo comentó. Promediando la presentación, [Martín⁷](#) repasaba algunos riesgos asociados a implementar un sistema [NoSQL⁸](#). Decía que es necesario tener en cuenta el costo capital (Capex), el costo operativo (Opex), si el sistema es propietario o si es Open Source y, por último, si existe gente capacitada en esta tecnología o si es necesario atraer a jóvenes promesas para capacitarlas. Inmediatamente, relacioné esto último a un tema actual, complejo y crítico, y que tiene que ver con la poca gente especializada que existe en el ámbito de la Informática. Lo digo en relación a la necesidad actual de profesionales informáticos y cómo de ellos depende el desarrollo de una “identidad tecnológica” como país. Claramente, esto no es algo nuevo y en estos enlaces hay algunas muestras de ello:

[Estudiar Computación⁹,](#)
[Las posiciones de trabajo más difíciles de cubrir¹⁰,](#)
[La escasez de profesionales¹¹, y](#)
[Hay que estudiar informática¹²](#)

⁶<http://www.cs.unm.edu/~cris/>

⁷<http://www.cs.unm.edu/~cris/>

⁸<http://en.wikipedia.org/wiki/NoSQL>

⁹<http://www.estudiarcomputacion.gob.ar/>

¹⁰<http://www.infobae.com/notas/723564-Las-posiciones-de-trabajo-mas-dificiles-de-cubrir.html>

¹¹<http://ernestokiszkurno.blogspot.com.ar/2013/05/la-escasez-de-profesionales.html>

¹²<http://ernestokiszkurno.blogspot.com.ar/2010/10/hay-que-estudiar-informatica.html>

Como conclusión, está claro que para desarrollar nuevas tecnologías e innovar, es necesario tener este **talent pool** impulsado por la “profesionalización” de la Informática como disciplina y por la necesidad de definir esta “identidad tecnológica”. Es también nuestra responsabilidad promover esta especialización.

Imagen: <http://www.insightfororganisations.co.uk/talent-management/>¹³

Tengo un problema

La técnica del **patito de goma**¹⁴ (reformulada)

Muchas veces nos pasa que nos bloqueamos cuando tenemos que resolver un problema (en este caso, laboral). No podemos avanzar, nos volvemos menos productivos, incómodos y no disfrutamos de nuestro trabajo. Para evitar esto, lo que la técnica propone es, simplemente, contarle a alguien lo que nos pasa. Luego, en medio del proceso, nos daremos cuenta de lo que está fallando (eventualmente) y nos ayudará a resolver el tema que nos aqueja. Pero tiene un problema, hay que molestar a otra persona para que nos atienda. Es aquí dónde aparece el Patito de Goma. Cuando estés bloqueado usá el patito, contale lo que te ocurre y verás cómo, en muchas ocasiones, te ayudará.

Contado de esta manera, tiene algo que me hace ruido y que contradice el “espíritu” de este blog: atenta contra los ambientes colaborativos. Por lo tanto, mi propuesta es:

- Si tenés un problema que no podés resolver y te está bloqueando: 1. Buscá un compañero para contárselo.
- Si no tenés más remedio y no tenés un compañero a mano: 2. Buscá una Patito de Goma.

Tengo un problema (remasterizado)

Luego del éxito sin precedentes del artículo anterior (¿?) y de algunos de los comentarios que me llegaron, quise hacer una remasterización del algoritmo de la técnica con el *feedback* que obtuve.

Si tenés un problema que no podés resolver y te está bloqueando, entonces: Si trabajás remoto: 1. Buscá un compañero en el chat para contárselo; Sino: 2. Contáselo a un compañero en persona; Si no tenés más remedio y no tenés un compañero a mano: 3. Buscá un Patito de Goma y/o cualquier otro; En otro caso: 4. Googlealo :)

Equivocarse

Leí este interesante [artículo¹⁵](#) de Gustavo Bonalde¹⁶, titulado **¿Siempre debemos fallar para lograr algo?**

¹³<http://www.insightfororganisations.co.uk/talent-management/>

¹⁴http://developerscookbook.blogspot.com.ar/2010_11_01_archive.html

¹⁵<http://gbonalde.blogspot.com.ar/2014/01/siempre-debemos-fallar-para-lograr-algo.html>

¹⁶<http://about.me/gbonalde>

Transcribo aquí una frase que me parece interesante:

“Es muy cierto que en cada falla tenemos una oportunidad de mejora, de aprendizaje, pero hay ocasiones donde no hay espacio para fallar.”

Y es verdad. Además, en aquellas ocasiones en las cuales no podemos fallar, y lo hacemos, seguro el “golpe” será más grande. Pero, a pesar de ellos, **seguimos aprendiendo**.

Para concluir, en cualquier profesión, trabajo, carrera, el aprendizaje es continuo y eso es lo mágico. Sólo hay que estar predispuesto.

No escribirás con errores ortográficos

Recuerdo que cuando estaba en la escuela primaria (6to grado), tenía una maestra que en una de las clases semanales de Lengua, nos tomaba un dictado. La dinámica era la siguiente: te dictaba 20 palabras elegidas al azar y, si tenías menos de 6 o 7 errores ortográficos, aprobabas. A medida que iba pasando el tiempo, la complejidad de las palabras se incrementaba y, para la calificación final de la materia, se ponderaba también cuántos dictados habías aprobado. Si bien era un examen y tenía un cierto nivel de stress, hoy en día, agradezco haber tenido la posibilidad de entrenar mi ortografía desde chico.

Esta anécdota es para introducir un tema que me parece muy importante, por no decir crítico. Digamos que podríamos seguir hablando sobre **skills blandas, duras¹⁷**, etc. de un profesional (en este caso, informático, tester, developer, no viene al caso). Sin embargo, pienso que antes de entrenar cualquiera de estas habilidades (que obviamente son también importantes), tenemos que estar seguros de escribir sin errores de ortografía. En resumen, antes de entrenar cualquier otra habilidad, procura escribir sin errores de ortografía.

Les dejo algunos tips que me ayudan a mejorar mi redacción:

- Leer, leer, leer.
- Consultar la **RAE¹⁸** o un diccionario físico.
- Tener a mano reglas de ortografía y gramática básicas.
- Usar sinónimos.
- Ejercitarse la escritura.
- Seguir en **Twitter¹⁹** a **@DelCorrector²⁰** (leer también su blog **El Santo de la Pluma²¹**)

Por último, no sé si a ustedes les pasa, pero me molesta leer un texto que tiene errores. Además, tiene consecuencias varias. Por ejemplo:

¹⁷<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>

¹⁸<http://www.rae.es/>

¹⁹<https://twitter.com/>

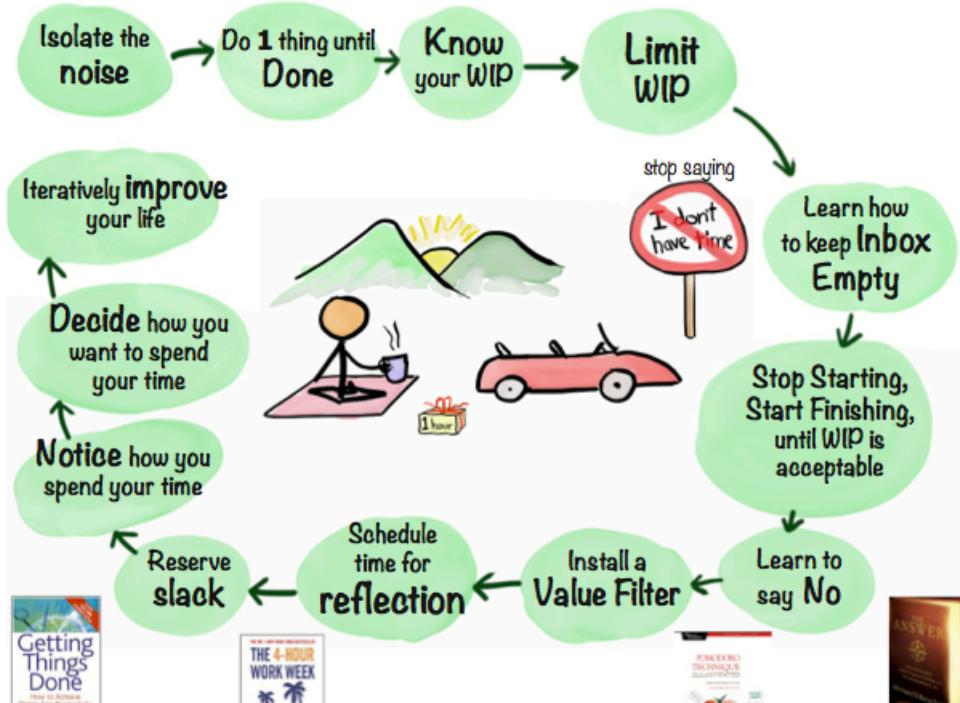
²⁰<https://twitter.com/DelCorrector>

²¹<http://elsantodelapluma.blogspot.com.ar/>

- Si es un informe, pierde toda seriedad al momento de localizar el primer acento mal puesto.
- Si es un bug, tu reputación de buen tester está en duda.
- Si es un chat, sms o mail informal, puede obviarse, de vez en cuando, una corrección no viene mal.
- Si es un CV, estás out.
- Si es una chica que tiene problemas con su ortografía, pierde inmediatamente todo su encanto.

Stop Starting, Start Finishing

Hace tiempo que encontré este slide de [Henrik Kniberg²²](#). Traté de implementarlo (y aún lo intento) pero algunas cuestiones son un poco más complejas. En el día a día, uno tiende mucho al *multitasking* y a salir corriendo atrás de las cosas que deberían haberse hecho para ayer.



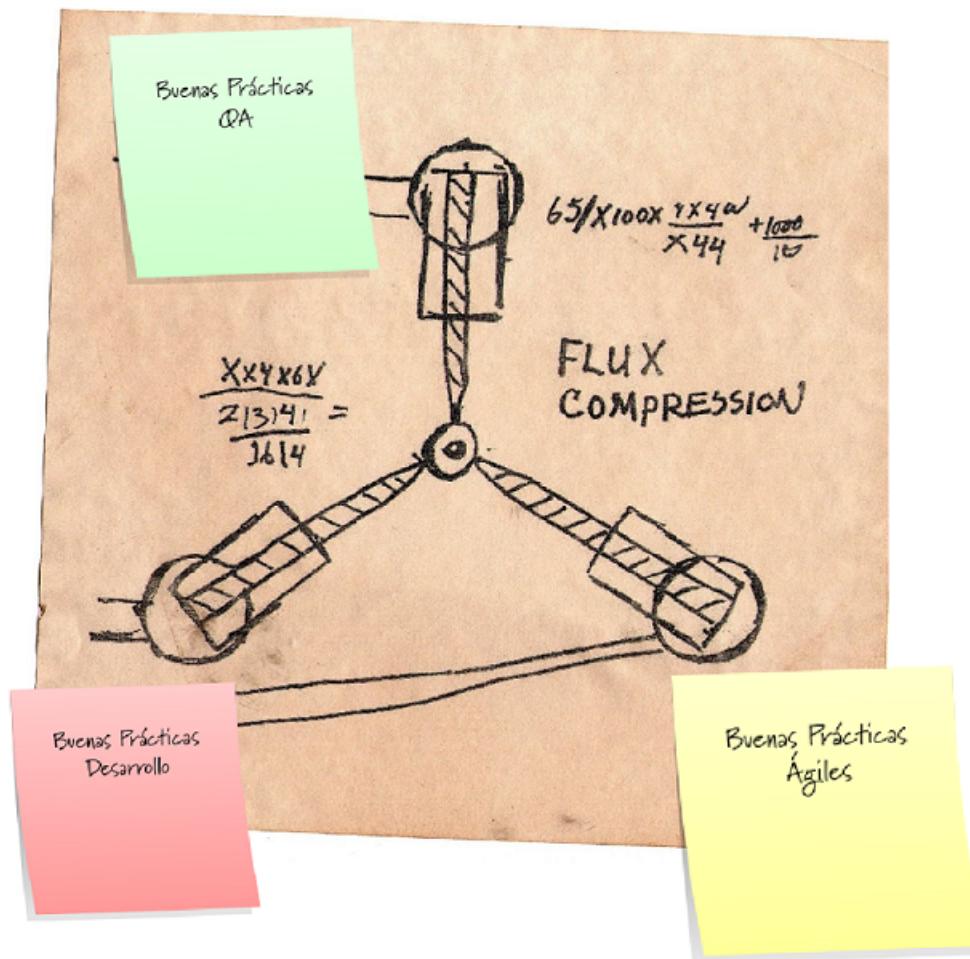
Stop Starting Start Finishing

En fin, admiro a la gente que sabe decir NO y que puede tener su WIP controlado...

²²<http://blog.crisp.se/author/henrikkniberg>

El condensador de flujo para nuestros proyectos

Sin bien la película no aclara cómo funciona exactamente, el condensador de flujo consiste en una caja con tres pequeñas lámparas incandescentes²³ centelleantes (nuestras prácticas QA, ágiles y de desarrollo), colocadas en forma de “Y”, y situada detrás entre los asientos de la máquina del tiempo. Cuando el automóvil (nuestro proyecto) se aproxima a una velocidad de 88 millas por hora (140 km/h), la luz que emiten las lámparas del condensador de flujo destellan con más rapidez, hasta emitir una luz constante (flujo constante de prácticas). Entonces el condensador se activa posibilitando el viaje en el tiempo.



El Condensador de Flujo

Si logramos incorporar buenas prácticas a nuestro equipo y a nuestros procesos a medida que el proyecto alcanza 88 millas por hora, vamos a viajar al tiempo del “buen” software. Es el espacio/tiempo donde habitan los productos de calidad, aquellos que realmente aportan valor y desarrollados por equipos altamente motivados y *skilled*.

²³http://es.wikipedia.org/wiki/L%C3%A1mpara_incandescente

En cambio, si no logramos incorporar estas buenas prácticas a nuestro equipo y a nuestros procesos, viajaremos al tiempo del software de baja calidad, con equipos estresados y con poco valor para aportar.

Nota: No creo que existan las “mejores prácticas” ya que su aplicación efectiva depende 100% del contexto en el cual serán aplicadas. Prefiero usar el concepto de “buenas prácticas” e involucrar el concepto de “contexto”. Algunas prácticas pueden ser buenas para un determinado contexto y no tanto para otro.

No revisaré mails al inicio del día

Aquí les dejo la justificación científica de por qué es interesante tratar de implementar el lema “[Stop Starting, Start Finishing](#)”²⁴.

El video “*La razón sobrevaluada*”, de [Estanislao Bachrach](#)²⁵ en la [TEDxRíoLimay](#)²⁶ del 2012 (en Septiembre de 2013 es la segunda edición), deja algunas perlitas:

1. Revisar mails al inicio del día consume mucha energía.
2. Dejar las actividades para las cuales necesitamos 100% de atención y son críticas para cuando estamos más descansados.
3. Priorizar
4. Visualizar
5. Escribir nuestros proyectos/ideas/compromisos para no tenerlos en nuestra cabeza constantemente.
6. Querer ser más productivos, nos hace improductivos.

Limitar nuestro WIP en pos de tomar mejores decisiones.

²⁴<http://germanbraun.blogspot.com.ar/2013/07/stop-starting-start-finishing.html>

²⁵<http://www.linkedin.com/in/estanislaobachrach>

²⁶<http://www.tedxriolimay.com.ar/>

Testing

Hacer testing es también capturar conocimiento

Varias veces he escuchado la siguiente frase: “Los desarrolladores tiene un pensamiento constructivo y los testers tienen un pensamiento destructivo”. Esta falacia se está derrumbando con el paso del tiempo (por suerte). Los testers también construyen, nada más ni nada menos que calidad. Ernesto Kiszkurno²⁷ ya habló de esto²⁸ en su blog.

En esta dirección, la dependencia de la gente en el software es cada vez mayor y, por lo tanto, este tiene que ser cada vez más confiable. Debe resolver el problema para el cual fue construido. Leandro Caniglia²⁹ siempre me dice que para él, construir software es capturar conocimiento y que los testers son el complemento indispensable en la cadena de valor. Es decir, los testers también participan en la captura del conocimiento tan importante a la hora de modelar dominios reales y, por lo tanto, complejos.

En conclusión, es hora de integrar definitivamente a los testers y a los developers para que ambos colaboren desde las fases tempranas de un producto. El testing NO debe ser una etapa en el ciclo de construcción de software sino que debe ser una actividad parte del criterio de “Done”.

“Tested is part of ‘Done’”

Testing Agile: conceptos y principios

Estoy leyendo el libro “Agile Testing: A Practical Guide for Testers and Agile Teams”³⁰ de Lisa Crispin³¹ y Janet Gregory³². Es interesante, ya que plantean el *Testing Agile* como:

- 1) Un testing **business-facing**, es decir, un claro enfoque en las funcionalidades y características deseadas por los expertos del negocio, sin perder de vista el espíritu crítico y objetivo propio del testing.
- 2) Una práctica que involucra a todos los miembros de un equipo, trabajando con una clara orientación a la calidad y hablando en un lenguaje común. En este contexto, el rol del tester es importante desde el inicio del desarrollo, en contraposición a posturas más clásicas en las cuales el testing es hecho hacia el final de un largo ciclo de desarrollo, con poco injerencia en el producto final

²⁷<http://ernestokiszkurno.blogspot.com.ar/>

²⁸<http://ernestokiszkurno.blogspot.com.ar/2012/01/frases-hechas.html>

²⁹<http://germanbraun.blogspot.com.ar/2013/01/desarrollar-software-6-preguntas.html>

³⁰<http://www.amazon.com/Agile-Testing-Practical-Guide-Testers/dp/0321534468>

³¹<http://www.linkedin.com/pub/lisa-crispin/0/20a/884/>

³²<http://ca.linkedin.com/in/janetgregory>

y enfocado, únicamente, en cumplir los requerimientos iniciales. Lisa y Janet definen este enfoque como **Whole-Team**.

Es claro que este nuevo concepto necesita de algunos **cambios culturales** y de nuevas **skills** de todas los involucrados en el proceso de desarrollo, *managers*, programadores, testers y clientes. Es un camino que puede ser largo o no según el contexto y las personas, sin embargo, podemos comenzar proveyendo soluciones prácticas que apliquen **valores ágiles³³** y promuevan sus principios.

Por último, Lisa y Janet proponen los siguientes 10 principios que son importantes para un tester agile y que voy a ir abordando en diferentes artículos:

- (1) Provide continuous feedback
- (2) Deliver value to the customer
- (3) Enable face-to-face communication
- (4) Have courage
- (5) Keep it simple
- (6) Practice continuous improvement
- (7) Respond to change
- (8) Self-organize
- (9) Focus on people
- (10) Enjoy

Hacer testing es también capturar conocimiento, otra vuelta

El post anterior³⁴ sobre este tema hizo bastante ruido. Recibí muchos comentarios/opiniones al respecto y me pareció interesante dar “otra vuelta”. En realidad, esta vez es para dividir en un par de *tracks* los comentarios que recibí y “generar más conocimiento”.

- **Testing Agile:** hace un tiempo escribí³⁵ sobre los principios ágiles propuestos en este libro³⁶, y efectivamente, de alguna manera, la idea expuesta en el post tiene implícitos estos principios.
- **Cualquiera puede hacer Testing:** sinceramente, me cuesta creer en esto. He trabajado con gente que no es del palo y han tenido un excelente desempeño. Sin embargo, hay cuestiones técnicas, de análisis, gestión, etc. en las cuales hay una clara ventaja del informático. Seguramente, en testing funcional es más factible, pero si queremos ir en esta dirección³⁷ deberíamos apuntar a gente informática, sin dudas. En otro momento, retomaré el tema.

³³<http://agilemanifesto.org/>

³⁴<http://germanbraun.blogspot.com.ar/2013/07/hacer-testing-es-tambien-capturar.html>

³⁵<http://germanbraun.blogspot.com.ar/2013/01/testing-agile-conceptos-y-principios.html>

³⁶<http://www.amazon.com/Agile-Testing-Practical-Guide-Testers/dp/0321534468>

³⁷<http://germanbraun.blogspot.com.ar/2013/08/talent-pool.html>

- **El Testing no puede automatizarse completamente:** No puedo estar más de acuerdo con esta afirmación. No hay dudas. Lo que sí surge aquí es, ¿qué automatizar y qué no?, ¿qué herramientas son las adecuadas?
- **Un fail como puerta de entrada hacia más conocimiento:** Un bug permite a los desarrolladores explorar escenarios, situaciones, y código aún no explorado y con un fuerte impacto (potencial) en la funcionalidad de software. Una de las maneras más efectivas de capturar conocimiento en testing es a través de un bug.

Para terminar, no deja de sorprenderme la cantidad de comentarios y de gente que tiene las mismas inquietudes. Los lectores son como los testers, ayudan a generar más conocimiento en este blog!

Trascribo aquí algunos de los comentarios.

“Interesante nota. No considero que los tester tengamos pensamiento destructivo, y va mi opinion desde la óptica de developer también ya que he tenido la suerte de ser ambas cosas. Se trata de un punto de vista diferente, no de construir o destruir, de ver las cosas desde otro ángulo y por eso coincido que debe existir una alianza entre ambos, lo que sería sin duda alguna, una dupla mas que potente!” Vero Puymalié

“El tester adopta otro punto de vista y ese es precisamente su aporte fundamental. “Point of view is worth 80 IQ points” (Alan Kay), no es una frase tirada al azar, es una idea central del pensamiento científico y creativo. El tester es un interlocutor calificado que suscita un mayor grado de conciencia y entendimiento. Un test que falla es una puerta que se abre a un mundo desconocido que tienta nuestra curiosidad y despierta nuestro interés.” Leandro Caniglia

“Es muy cierto, el tema es - y lo puedo decir con propiedad ya que he estado en ambos bandos - es que los de desarrollo ven a los testers como enemigos potenciales y no como la parte que complementa su trabajo ya que ellos (los de desarrollo) son los únicos con el conocimiento y siempre tienen la razón y eso es lo que no puede continuar ocurriendo... Hay que generar un cambio dentro de los equipos de desarrollo e involucrar a los testers como parte de la solución y no del problema y hay que hacer mucho énfasis en que sin las pruebas de rigor el desarrollo no está completo.” Augusto Cardozo Martinez

“...Llevo solo unos meses como QA y la primera impresión que tuve en una reunión fue que ambas áreas son casi enemigas. Los desarrolladores sienten que nosotros estamos para refregarles los errores en su cara, pero no es así, solo queremos que nuestro cliente obtenga un software de calidad, ojalá con margen de error cero y sentirnos satisfechos por entregar un servicio 100% de calidad y confiable.” María Ramos Campos

“Ciento es que los testers caemos habitualmente en ver únicamente lo negativo de los bugs, cuando muy probablemente sean de lo mejor que tenemos para tomar la temperatura de nuestro modelo de testing. No encontrar bugs no quiere decir que no estén ahí, lamentablemente, por ello tenemos que tratarlos como lo que son y pueden ser, una herramienta de mejora.” Mauri Edo

Basta de “Dev vs QA”



imagen

Uso [Feedly³⁸](#) para gestionar los post de los blogs que sigo. Diariamente, hago una selección de artículos para leer en el momento y otros los guardo para chusmeárselos luego. Esta selección la hago por categoría, nombre de autor (algunos leo con más frecuencia que otros) y también hago selección por título.

En este contexto, todos los días me encuentro con que aún se sigue escribiendo (bastante) sobre la “rivalidad”, “diferencia”, importancia de los testers por sobre los developers o viceversa.

A esta altura del partido, creo que este debería ser ya un tema superado. Por esta razón, es que he decidido no leer más artículos con estos contenidos. Títulos tales como “Developers vs. Testers”, “¿Por qué los programadores ‘odian’ a los testers?”, “El espíritu destructivo de los testers” están para mi obsoletos. Opino que ambos (devs y testers) deberían estar bajo un mismo rótulo: *software makers*, o simplemente *software developers*. Ya no tiene mucho sentido volver sobre lo mismo y debería ser un tema superado en la comunidad del software. Hay que pasar al siguiente nivel de la discusión.

³⁸<http://feedly.com/>

Principios esenciales para Automatización de Pruebas Funcionales

La última [Testing Experience³⁹](#) tiene un interesante artículo sobre principios para tener en cuenta al momento de automatizar tests funcionales. Los quería compartir aquí ya que me parece importante tenerlos bien presentes.

- **Primero, diseñar los tests.** Evitar la creación de tests “on the fly”. Nuestra suite de tests debe crecer orgánicamente.
- **No automatizar todo.** Ciertos tests pueden ser fácilmente ejecutados en forma manual, evitando así el alto costo de automatización.
- **Escribir tests cortos.** En situaciones en las que uno o más tests fallen, cualquier miembro del equipo debe ser capaz de *trackear* la causa del error. Es recomendable usar [BDD \(Behaviour-Driven Development\)](#)⁴⁰
- **Crear tests independientes.** Evitar el acomplamiento y aumentar la cohesión de los tests.
- **Enfocarse sobre su readability.** El código fuente de cada tests debería estar *self-documenting*. Comentarios en el código deben ser evitados.
- **Tests deben ser rápidos.** El testing funcional automatizado debe ser un indicador rápido de la calidad de la aplicación. Además, en un ambiente de [continuos delivery⁴¹](#), el tiempo de ejecución debe ser de unos pocos minutos.
- **Crear tests resistentes al cambio.** Quizás esta es una de las principales desventajas de los tests funcionales automatizados. Tests debería verificar sólo funcionalidad. Dependiendo del contexto también puede ser posible aplicar [DDT \(Data-Driven Testing\)⁴²](#)
- **Los tests automatizados no pueden reemplazar a los humanos.** Este no debe ser el único testing que se ejecuta. Otras técnicas, con la intervención de los testers humanos, deben ser aplicadas sobre el software en pos de generar más conocimiento.

¿Por qué los testers necesitan aprender continuamente?

Últimamente estoy leyendo mucho a Lisa Crispin y Janet Gregory. Esta vez, encontré el artículo [“Why Testers and QA Engineers Need to Learn Continuously?”⁴³](#). Me parece atinado verter aquí algunos de sus conceptos ya que hacer un tiempo venimos diciendo que los testers trabajan con conocimiento⁴⁴.

³⁹<http://www.testingexperience.com/>

⁴⁰<http://dannorth.net/introducing-bdd/>

⁴¹<http://www.amazon.com/dp/0321601912?tag=continidelive-20>

⁴²http://en.wikipedia.org/wiki/Data-driven_testing

⁴³<http://www.agileconnection.com/article/why-testers-and-qa-engineers-need-learn-continuously>

⁴⁴<http://germanbraun.blogspot.com.ar/2013/07/hacer-testing-es-tambien-capturar.html>

- 1) Expandir tu conocimiento y tus capacidades te posibilitará incrementar tus oportunidades, tanto dentro como fuera de tu organización.
- 2) Testers quienes eligen aprender, abrir su mente y probar nuevas cosas, son más valiosos para sus equipos. Los buenos testers entienden tanto el negocio como los aspectos técnicos de su producto.
- 3) El aprendizaje reduce el stress. Invertir tiempo en adquirir nuevo conocimiento permite aumentar la confianza en cómo hacer tu trabajo y, además, hacerlo correctamente.
- 4) Cuando tu compartes nuevo conocimiento con tu equipo, “desafías” a tus compañeros a pensar nuevas y mejores maneras de hacer las cosas.
- 5) La sabiduría que da la experiencia combinada con el aprendizaje, el perfeccionamiento de tus *skills* y la constante actualización sobre las nuevas ideas que surgen desde la industria (y la academia), no sólo te vuelven más “vendible” sino que también te hacen más valorable dentro del equipo en el que trabajas.
- 6) El aprendizaje continuo te da la posibilidad de formar parte de una comunidad de pensadores que comparten experiencias e ideas que pueden crecer orgánicamente.

Más allá de todos estos puntos, debe existir una motivación intrínseca orientada a comprender las **responsabilidades del tester⁴⁵** y sus **habilidades⁴⁶** y a desafiar el **status quo⁴⁷**.

Never lose a BUG again

Sabios consejos para los testers⁴⁸:

- Controlá tus tests.
- Compartí tus *bugs* tal como los encontrás.
- Capturá tu pantalla, adjuntá al incidente y seguí trabajando. Los *developers* verán la aplicación exactamente como estaba cuando encontraste el *bug*.
- Eliminá las excusas: no digas más “Funciona en mi PC”.

What's a Tester without a QA Team? (hacia un entorno ágil)

Luego de escribir sobre “**Así comienza el camino...**”⁴⁹, encontré un interesante artículo escrito por Lisa Crispin y Janet Gregory, titulado “**What's a Tester without a QA Team?**”⁵⁰. Quería remarcar una

⁴⁵<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

⁴⁶<http://germanbraun.blogspot.com.ar/search?updated-max=2013-04-05T19:42:00-03:00&max-results=7&start=14&by-date=false>

⁴⁷<http://germanbraun.blogspot.com.ar/2013/03/responsabilidad-cambios-y-motivacion-mi.html>

⁴⁸http://go.cloudshare.com/LinkedinAds_QA.html?source=Linkedin&camp=QA&medium=CPC&term=NeverLoose&content=&term=

⁴⁹<http://germanbraun.blogspot.com.ar/2013/12/asi-comienza-el-camino.html>

⁵⁰<http://www.agileconnection.com/article/what%E2%80%99s-tester-without-qa-team>

serie de bullets que me parecen interesantes para llevar a la práctica cuando un tester transiciona de un esquema tradicional a uno ágil.

- **Preparándose para el éxito.** Inducir y dar soporte para que la persona no se aísle. Ahora, en este nuevo enfoque, se debe reportar a otras personas, otros roles y los testers deben aprender a colaborar con otros *software makers*⁵¹ del equipo.
- **El enfoque “Whole Team”.** En los equipo ágiles exitosos, todos los miembros tiene un claro enfoque en la calidad y son responsables de ella. La buena noticia para los testers es que ahora ellos van a estar involucrados en el proyecto desde el principio. Su tarea comienza con los clientes o product owner definiendo los tests de aceptación y continua con la revisión de user stories junto con los developers.
- **Comunidad de Práctica de Testing.** Construir comunidades de práctica que faciliten la integración del equipo. El fin principal es compartir conocimientos y experiencias. Esta comunidad puede tener un test manager (responsable del equipo de testing en el enfoque tradicional) quien sea el encargado de asegurar la inducción y el soporte a los testers. El equipo debe también estar orientado a *hacer comunidad*.
- **Compartir Conocimiento.** Cuando el equipo tiene el sentido de comunidad, se genera un núcleo de conocimiento muy poderoso que favorece el intercambio. Proactivamente, los integrantes comienzan a intercambiar conocimiento de manera desinteresada y totalmente natural.
- **Ampliar tus horizontes.** Todos los *software makers*⁵² deben estar enfocados en el aprendizaje continuo. Cada vez que una nueva funcionalidad es pensada y desarrollada, se produce una explosión de nuevo conocimiento que involucra y retroalimenta a todo el equipo.

Habilidades de un tester

Según la RAE⁵³, habilidad es la capacidad y disposición de una personal para hacer algo. Generalmente, esto es resultado de experiencia, entrenamiento y, en algunos casos, sólo es una capacidad natural.

Un tester debe tener ciertas habilidades para desarrollar su trabajo, algunas **habilidades técnicas** (también las vamos a llamar *hard*) para ejecutar cuestiones básicas de su trabajo y otras **habilidades soft** que tienen que ver con su enfoque hacia el trabajo. Estas últimas van más allá de los límites de la profesión y alcanza otros aspectos de la vida. Son habilidades que hacen a su comunicación interpersonal, negociación y resolución de conflictos, efectividad y creatividad, por nombrar algunas.

A partir de esta diferenciación y teniendo en cuenta la importancia de cada una, podemos definir 3 grupos:

⁵¹<http://germanbraun.blogspot.com.ar/2013/09/basta-de-dev-vs-qa.html>

⁵²<http://germanbraun.blogspot.com.ar/2013/09/basta-de-dev-vs-qa.html>

⁵³<http://www.rae.es/rae.html>

1. Aquellas profesiones que necesitan de las habilidades técnicas y pocas habilidades *soft*.
2. Aquellas profesiones que necesitan ambas habilidades
3. Aquellas profesiones que necesitan mayormente habilidades *soft*, pero no así de tantas *hard*.

Pienso que la profesión de tester está contenida en el segundo grupo y, a continuación, justifico mi elección.

Las habilidades técnicas son aquellas que implican “meterse en el barro”, hacer una estrategia de pruebas, definir/programar y ejecutar casos y generar los respectivos reportes e informes. También, incluyo aquí la aplicación de metodologías para la gestión de los proyectos, entendimiento del dominio/negocio, procedimientos generales y adopción de estándares.

Por último, dejé para el final las habilidades *soft* ya que son las más difíciles de observar, cuantificar y medir y, por lo tanto, quiero hacer especial hincapié sobre ellas. Esto no hace más importantes a estas habilidades por encima de las técnicas (de hecho, un tester no sobreviviría sin habilidades técnicas). El tema pasa por como las habilidades pueden ser adquiridas. Las *hard*, *pueden* entrenarse con mayor facilidad ya que en cierta medida se aprenden en la escuela, Universidad, cursos, etc. Mientras que las habilidades *soft* están relacionadas a los patrones de conducta de las personas y, en general, se mejoran/modifican (y entran también) haciendo camino al andar.

A continuación detallo las habilidades *soft* que un tester debería tratar de atender, mejorar y entrenar para ser un tester “completo”:

1. Disciplina y perseverancia
2. Meticulosidad
3. Comunicación (oral y escrita)
4. Curiosidad
5. Observación (fundamental!)
6. Priorización de esfuerzo y gestión del tiempo
7. Actitud positiva
8. Pensamiento negativo
9. Pensamiento crítico

Quizás, en estas últimas, esté el ***quid de la cuestión...***

Fuentes:

- [1] [Stickyminds⁵⁴](http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=6752)
- [2] [Así no se hacen las cosas⁵⁵](http://ernestokiszkurno.blogspot.com.ar/2010/02/tester-mindset.html)
- [3] [Cartoon Tester⁵⁶](http://cartoontester.blogspot.com.ar/)

⁵⁴<http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=6752>

⁵⁵<http://ernestokiszkurno.blogspot.com.ar/2010/02/tester-mindset.html>

⁵⁶<http://cartoontester.blogspot.com.ar/>

Tester World-Class

Luego de haber hablando un poco sobre las responsabilidades⁵⁷ y habilidades⁵⁸ de un tester, encontré por ahí un tweet que hacia referencia al siguiente post [Becoming a World-Class Tester](#)⁵⁹. Me gustó lo siguiente.

Primero, la definición de Tester World-Class:

“My definition of a world-class tester is a person who is able to rapidly discover highly relevant information about the product, who makes the most use of any resource that is available to him/her, and who has the respect of people involved in a project. It is a person who can be trusted.”

Segundo, habla sobre las *skills* y *mindset* de un tester world-class de las cuales quiero rescatar 2 para agregar a la lista publicada [aquí](#)⁶⁰:

1. **Voluntad para aprender.** Testers trabajan con conocimiento. El conocimiento no es estático, por lo tanto, el aprendizaje constante es esencial para mejorar la interacción *humano-software*.
2. **Humor.** El humor ayuda a mantener la cordura, principalmente, en ambientes estresantes. También, ayuda a enfocarse sólo en lo que se quiere hacer.

La T de Testing

Continuo con los posts sobre habilidades. Esta vez, me gustó la idea de [T-Shaped Testers](#)⁶¹ ya que es un buen resumen de lo que comenzó [aquí](#)⁶² y luego siguió con esta [entrada](#)⁶³ y por último, [aquí](#)⁶⁴.

Imaginemos una letra T. La linea vertical de la letra representa las habilidades centrales y la experiencia de una persona. En el caso del testing, pueden cambiar según el contexto, la persona, su trabajo, etc. por lo tanto, cada uno piense las habilidades que le parezcan centrales para un tester.

⁵⁷<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

⁵⁸<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>

⁵⁹<http://www.ebaytechblog.com/2013/01/31/becoming-a-world-class-tester/>

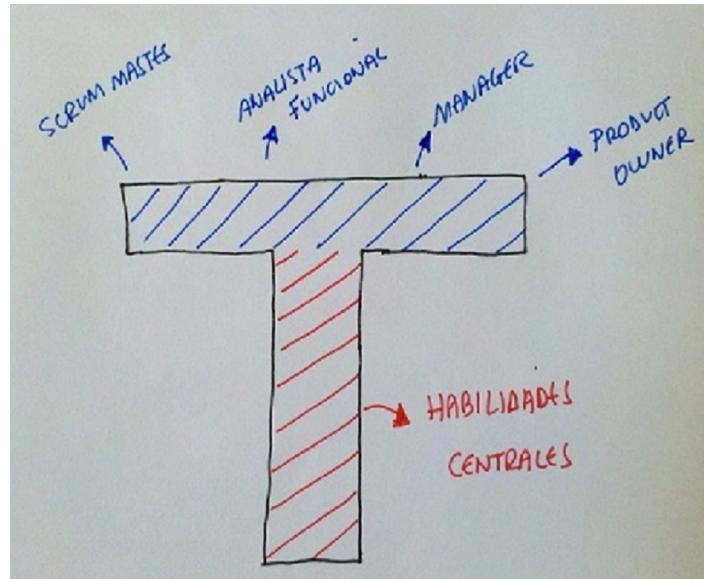
⁶⁰<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>

⁶¹[http://thesocialtester.co.uk/t-shaped-testers-and-their-role-in-a-team/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%2A+TheSocialTester+\(The+Social+Tester\)](http://thesocialtester.co.uk/t-shaped-testers-and-their-role-in-a-team/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%2A+TheSocialTester+(The+Social+Tester))

⁶²<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

⁶³<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>

⁶⁴<http://germanbraun.blogspot.com.ar/2013/03/tester-world-class.html>



La T de Testing

Por otro lado, la parte horizontal de la T presenta la capacidad de la persona para poder trabajar en múltiples disciplinas, usar sus habilidades centrales en otros dominios/roles y también de obtener nuevas desde otras áreas.

A modo de primera conclusión, claramente, aquellas personas T-Shaped pueden verse beneficiadas en su lugar de trabajo dado que pueden cumplir (o potencialmente) múltiples roles.

Sin embargo, quiero hacer un poco más de hincapié en testing y remarcar la importancia que tiene este enfoque en el contexto del testing de software tal como venimos comentando hace un tiempo en este blog.

1. Testers T-Shaped son necesarios en la práctica de [Testing Agile](#)⁶⁵. Testers involucrados en el producto y en dar valor desde el primer momento, desplegando sus habilidades en T para soporte en múltiples tareas y roles.
2. Testers T-Shaped son más flexibles y se adaptan más rápidamente al contexto y al dominio.
3. Testers T-Shaped comprenden sus [responsabilidades](#)⁶⁶.
4. Testers T-Shaped son [Testers World-Class](#)⁶⁷.

Testing no es sólo encontrar bugs

⁶⁵<http://germanbraun.blogspot.com.ar/2013/01/testing-agile-conceptos-y-principios.html>

⁶⁶<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

⁶⁷<http://germanbraun.blogspot.com.ar/2013/03/tester-world-class.html>

Mi responsabilidad como tester

Investigando sobre context-driven-testing, encontré un interesante artículo de Michael Bolton⁶⁸ en el cuál se trataba de definir este concepto mediante un diálogo entre James Bach⁶⁹ y Jerry Weinberg⁷⁰. Allí, Jerry dice dos cosas muy importantes:

- “*The first responsibility of a tester is to explore and challenge the constraints of the situation*”
- “*A tester is someone who knows that things could be different.*”

Es genial, es lo que estaba buscando hace mucho tiempo. Ambas proposiciones resumen lo que para mí es ser un tester y lo que le da sentido a esta profesión.

Encontrar el sentido a tu profesión/carrera/trabajo, preguntarte ¿por qué?, ¿para qué?, replantearte si estás en el lugar que querés estar y haciendo lo que hacés, hacerte cargo de tu carrera y salir de la “zona de confort”, también forma parte de tu vida, de “hacer el click”.

Una experiencia aplicando Kanban en Testing

Una experiencia aplicando Kanban en Testing [Parte 1] Hace unos días rearreglamos el tablero Kanban de un proyecto de testing en el que estoy actualmente trabajando. La experiencia fue interesante ya que debíamos explicar el tablero y los conceptos principales de esta metodología a integrantes nuevos del equipo. Esto nos obligó a repasar, hacernos preguntas, definir nuevos criterios y generar discusiones en el grupo.

Proyecto

Servicio: Testing Funcional

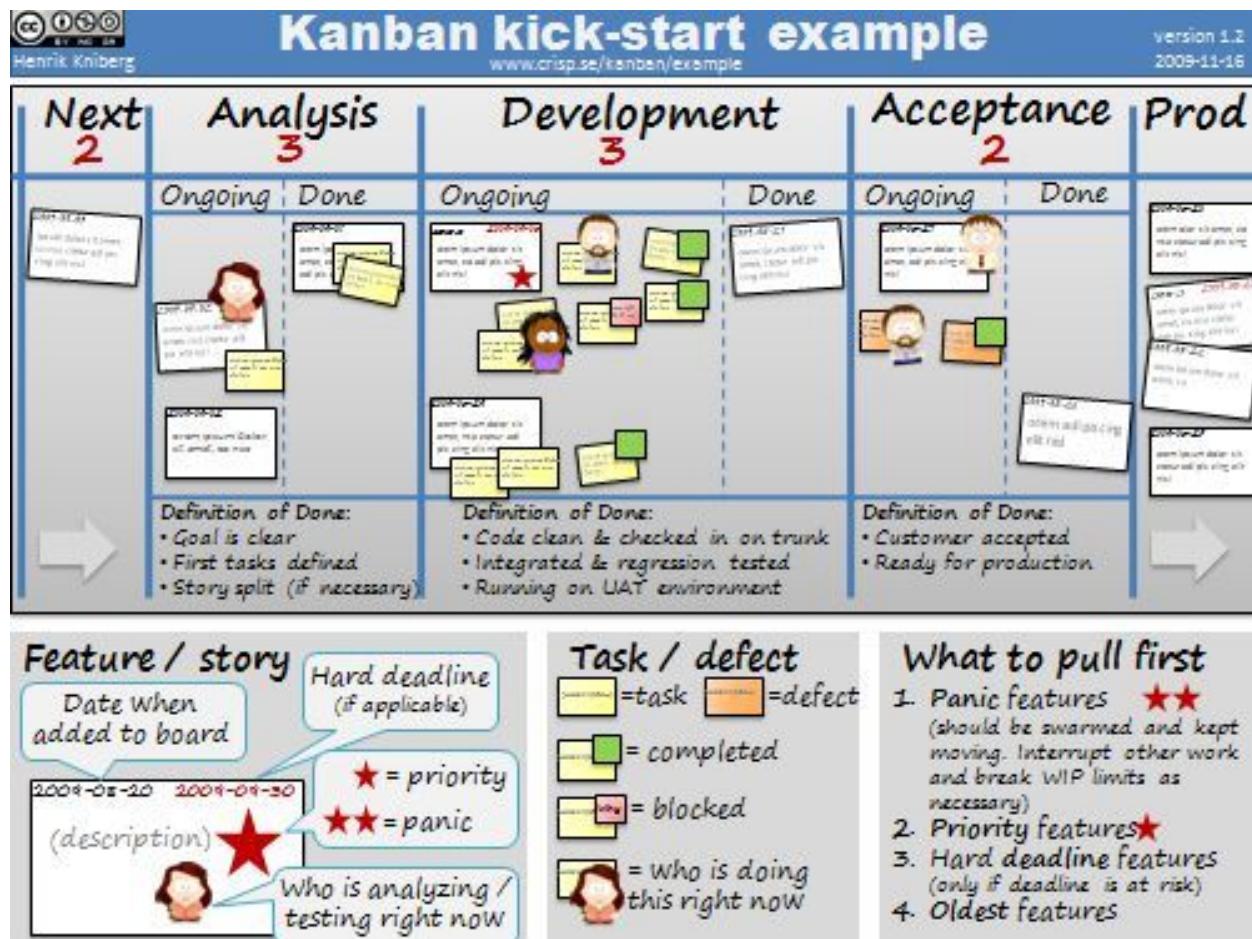
Equipo: 3 personas (1 diseñador de casos y 2 ejecutores)

Objetivo: Agregar valor al producto manteniendo el equipo asignado a tareas productivas

⁶⁸<https://twitter.com/michaelbolton>

⁶⁹<http://www.satisfice.com/aboutjames.shtml>

⁷⁰<http://www.geraldmweinberg.com/Site/Home.html>



Ejemplo Tablero Kanban

Implementado Kanban

Hicimos una reunión con el equipo en la que discutimos los siguientes puntos (notar que algunos pueden ser muy generales y otros quizás más específicos del proyecto):

1. ¿Qué es Kanban?

Como definición general, es una técnica que se usa para gestionar el proceso de desarrollo de software de una manera eficiente. Kanban es una palabra japonesa que puede traducirse como tablero visual o sistema de tarjetas.

2. ¿Cuáles son las reglas de la metodología?

- **Visualizar el flujo de trabajo:** dividir el trabajo en tareas/items/funcionalidades/necesidades, escribirlas en una tarjeta (post-it) y pegarlas en un tablero. Usar columnas etiquetadas con las etapas de nuestro proceso para indicar dónde, cada item, está en dicho proceso.
- **Limitar el trabajo en progreso (WIP):** asignar un límite explícito a cuántas asignaciones pueden estar en progreso en cada etapa del workflow.

- **Medir el “lead time”:** (tiempo promedio para completar un ítem) optimizar el proceso para hacer el lead time tan pequeño y predecible como sea posible.

3. ¿Para qué armamos un tablero? ¿Qué información nos brinda?

El tablero es la **foto** del proyecto. Necesitamos visualizar su estado actual, tareas en curso, manejo de asignaciones, limitaciones, “cuellos de botella”, problemas y mejoras para optimizar el proceso. Además queremos favorecer la autogestión del team, balancear necesidades e incentivar la participación, colaboración y expresión de todos sus miembros.

4. ¿Lo hacemos?

Las columnas etiquetadas con las etapas del proceso dependen del proyecto. En este caso, diseñamos el tablero de la siguiente manera:

- **Backlog:** lista de necesidades que deben ser testeadas, ordenadas por prioridad.
- **En Definición:** features en esta etapa están en definición de estrategia y diseño de casos de prueba.
- **En Cola de Ejecución:** los casos para la funcionalidad fueron definidos y están esperando la ejecución.
- **En Ejecución:** la ejecución de casos comenzó.
- **Finalizado:** la funcionalidad ya fue testeada (*)

5. ¿Cómo lo mejoramos?

- **Fila EXPRESS:** puede darse el caso de que nuevas funcionalidades tengan prioridad por sobre las planificadas inicialmente. Esto (a veces) implica dejar de trabajar sobre las que están en proceso y, por lo tanto, es necesario definir un segmento express por donde “circulen” estos ítems con mayor prioridad.
- **Fila BLOQUEADOS:** por aquí van las features que no pueden avanzar en nuestro proceso. Esto puede deberse a falta de documentación (bloqueado en definición de casos), funcionalidad no implementada (bloqueada esperando ejecución) o bugs invalidantes (bloqueado en ejecución). Además, este segmento funciona como segundo ciclo de pruebas con funcionalidades que pueden volver a priorizarse cuando son desbloqueadas.

Por último, luego de haber diseñado el tablero del proyecto, el equipo comenzó a discutir sobre cómo trabajar con el workflow definido, qué información debemos registrar para mantener trazabilidad, cómo medir, el work-in-progress y algunos criterios de aceptación (*).

Continuará...

Fuentes:

Foto⁷¹

⁷¹<http://blog.crisp.se>

Kanban Blog⁷²

Fuerza Tres (1)⁷³

Fuerza Tres (2)⁷⁴

Visual Management⁷⁵

“Kanban and Scrum: making the most of both” Henrik Kniberg y Mattias Skarin

Dar feedback continuo

Primer principio del *Testing Agile*⁷⁶

Como venimos diciendo hace un tiempo, en el contexto de un equipo ágil, el tester está involucrado en el producto desde los inicios. Desde bien temprano en la primer iteración. Muchos proyectos ágiles están “manejados” por *tests*, por lo tanto, el *feedback* juega un rol importante en ellos. En principio, el *feedback* comienza trabajando con el *product owner* o cliente para articular las *stories* mediante ejemplos y *tests*. Luego, los testers trabajan junto con sus otros miembros del equipo para generar escenarios ejecutables a partir de los requerimientos relevados. En consecuencia, la ejecución de estos *tests* es una fuente continua de *feedback* para el equipo. Por último, los testers pueden mostrar su progreso dando una vista global de *tests* creados/ejecutados/OK así como del cubrimiento funcional de sus pruebas. En cualquiera de estas situaciones, es una manera de ayudar a remover obstáculos y promover la cultura del trabajo colaborativo.

⁷²<http://www.kanbanblog.com/explained/>

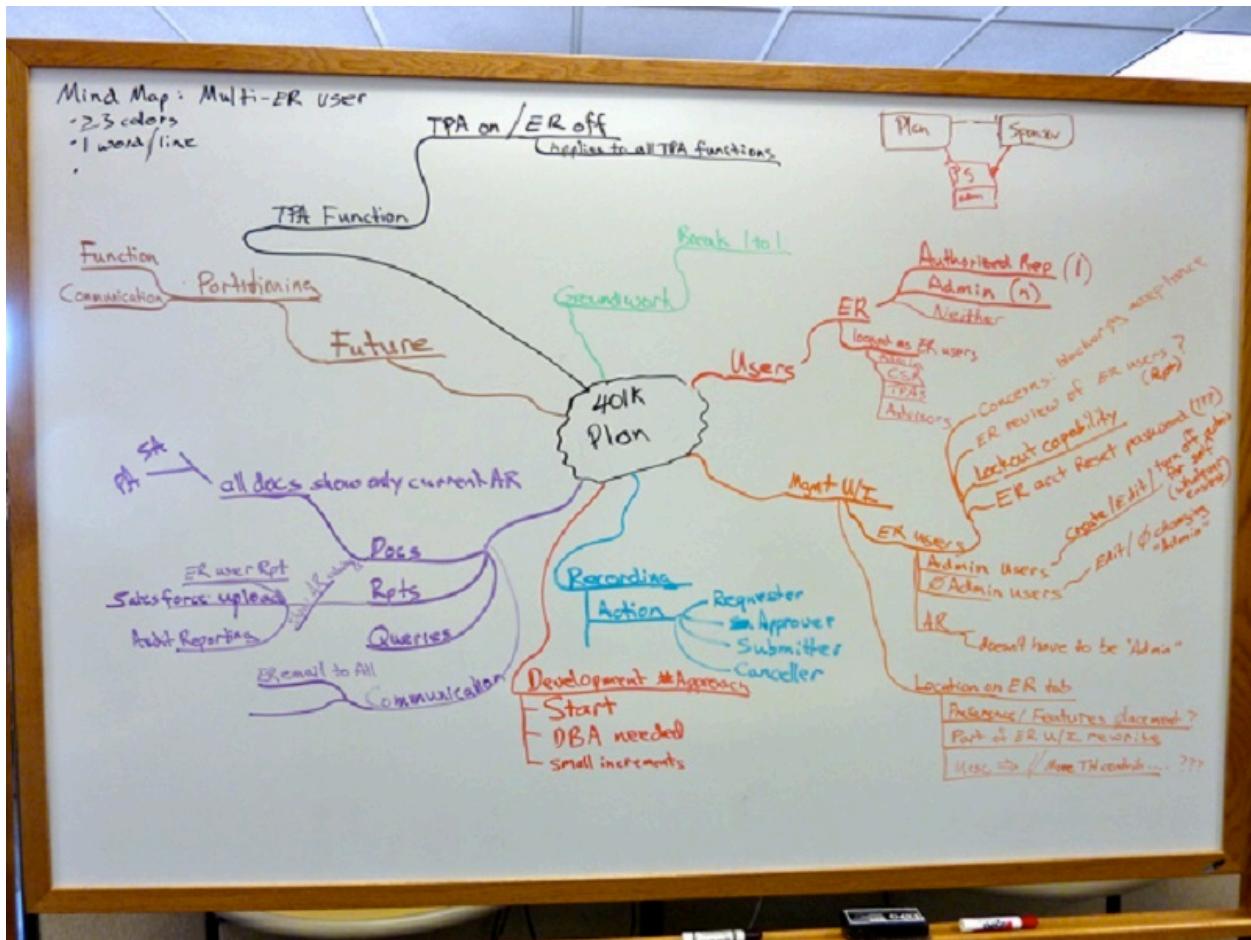
⁷³<http://www.fuerzatres.com/2011/07/trabajar-sin-estres-kanban.html>

⁷⁴<http://www.fuerzatres.com/2011/03/kanban.html>

⁷⁵<http://www.xqa.com.ar/visualmanagement/tag/kanban/>

⁷⁶<http://www.amazon.com/Agile-Testing-Practical-Guide-Testers/dp/0321534468>

Mind Mapping en Testing



Mind Map

En este último tiempo, en mi trabajo, hemos comenzado a usar mind maps para plantear estrategias y derivar pruebas funcionales. Nos parece una herramienta muy potente para tratar la complejidad del testing y del software.

Un **mind map** (o mapa mental) es una técnica gráfica que permite representar y relacionar ideas alrededor de una idea central. La característica principal, por la cual son muy utilizados, es que organizan la información de la misma manera que funciona nuestro cerebro, trabajando sobre sus hemisferios y creando diagramas que son fácilmente recordables.

Para hacer un mapa mental, hay que tener en cuenta lo siguiente:

1. Escribir la idea principal en el centro del diagrama.
2. Los temas importantes debe ser graficados como branches (o ramas), asociados a la idea principal. Cada nueva rama es agregada en el sentido de las agujas del reloj.

3. A cada rama debe asociarse una imagen o una palabra clave.
4. Los temas/conceptos de menor relevancia son incluidos en el mapa como subramas de las ramas principales.

Mind maps para estrategias y diseño de tests

Como ya he comentado, generar un buen plan de pruebas es complejo debido a que no sólo debemos tener en cuenta la funcionalidad a testear sino también cómo ésta se relaciona con otras y su impacto en el software. Aquí, la explosión de combinaciones y escenarios puede ser aún mayor.

Un mapa mental nos permitirá abordar todas estas cuestiones de una manera rápida y simple. Para esto, es esencial que el mind map sea desarrollado, como mínimo, de a pares. En caso de ser necesario se puede incluir en el grupo, a algún analista funcional o developer pero sólo para evacuar dudas funcionales puntuales y así mantener a salvo la objetividad de la estrategia.

Pasos:

1. Reunir al equipo.
2. Elegir un moderador.
3. Utilizar una pizarra o alguna herramienta software. Esto último es preferible para mantener la documentación. También es deseable tener un template para no tener que comenzar el mapa desde el scratch.
4. Escribir la funcionalidad a testear en el centro del mapa.
5. Agregar como ramas principales, las categorías para dividir y priorizar la funcionalidad (ej, ABM, seguridad, performance, datos, funcionalidades relacionadas, etc)
6. Empezando por la rama prioritaria, agregar las condiciones de cada prueba como ramas secundarias.
7. Marcar branches prioritarios con colores, números, imágenes.

Algunos beneficios que se obtienen cuando se utiliza esta técnica:

1. Inducción para todos los testers participantes y otros testers.
2. Mayor visibilidad sobre la funcionalidad, su relación con otras y sobre el software en general.
3. Flexibilidad ante cambios en requerimientos.
4. Incremento del coverage.
5. Identificación y priorización de escenarios importantes.
6. Soporte al ciclo de ejecución de pruebas.
7. Documentación.
8. Aumento de la creatividad del equipo.

En resumen, los mapas mentales son muy poderosos, flexibles y bondadosos, no sólo para testing, sino también para [tomar notas⁷⁷](#), hacer presentaciones, tomar decisiones y hasta gestionar tareas.

Fuentes:

[Mind Mapping⁷⁸](#)

[Quick Testing Tips⁷⁹](#)

[Imagen⁸⁰](#)

Testing + Pensamiento Lateral = “Testing Lateral”

El testing es un proceso intelectual desafiante y por ende requiere, entre otras cosas, de una dosis importante de creatividad. Esto se practica, se entrena.

Un tester que entrena sus habilidades e incorpora nuevas se diferencia claramente del resto. Todos en el equipo podríamos testear sin problemas, pero aquel con una mayor actitud y predisposición para responder ante las diferentes situación, incluso aquellas triviales, es quién llevará la delantera.

Pensamiento Lateral

En vacaciones, leí “[El Pensamiento Lateral](#)⁸¹” de Edward De Bono⁸². El libro es muy recomendable, y tiene una serie de técnicas que ayudan a “practicar” el pensamiento lateral además de incluir algunos ejercicios para tal fin.

De Bono afirma que usar pensamiento lateral no implica dejar de lado el pensamiento lógico/vertical, sino que es un complemento de este y que, además, nos ayuda a romper con ciertos modelos preestablecidos. Algo así como “barajar y dar de nuevo” nuestros modelos mentales. El pensamiento lateral de aplica en una fase anterior a la acción del pensamiento vertical. Se usa para reestructurar los enfoques de una situación determinada y las ideas que sirven de base. Luego estos enfoques e ideas pueden ser desarrolladas por el pensamiento vertical.

Me pregunto si será posible aplicar alguna de estas técnicas para entrenar y fortalecer las [skills de los testers⁸³](#), por ejemplo, en algún Dojo de Testing. Lo voy a intentar exponiendo técnicas y profundizando sobre este enfoque en sucesivas entradas. Espero que resulte algo novedoso e interesante como para experimentarlo.

Testing Lateral

Llámemos a este enfoque: “**Testing Lateral**”. Se trata, principalmente, de aplicar técnicas del pensamiento lateral para mejorar las habilidades de los testers. Este enfoque no intenta cubrir todo

⁷⁷<http://germanbraun.blogspot.com.ar/2013/01/mantenerse-creativo-2.html>

⁷⁸<http://www.mindmapping.com/>

⁷⁹<http://www.quicktestingtips.com/tips/category/mind-mapping/>

⁸⁰<http://lisacrispin.com/2011/01/09/mind-maps-for-theme-planning/>

⁸¹<http://www.amazon.com/s?search-alias=stripbooks&field-isbn=0060903252>

⁸²http://en.wikipedia.org/wiki/Edward_de_Bono

⁸³<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>

el rango de posibles habilidades que se consideren ya que para las habilidades que son técnicas o muy especializadas, se requiere de la aplicación de conocimientos adquiridos.

El testing lateral intenta trabajar sobre algunos aspectos *soft* de los tester, es decir, aquellas habilidades que hacen a su mindset, a su creatividad y su meticulosidad (algunas otras habilidades las detallamos [aquí⁸⁴](#)). Esto les permitirá considerar diferentes enfoques al momento de resolver cualquier situación que se presente.

Como notarán, el enfoque es simple. De ahora en más, sólo resta conocer algunas técnicas y ejercicios para ver si funciona. Las siguientes son algunas de las técnicas (un subconjunto de las propuestas por De Bono en su libro) que vamos a probar bajo este enfoque, siempre tratando de poner en práctica las habilidades de los testers:

1. Revisión de supuestos
2. Alternativas
3. Ejercicios de dibujo
4. Fraccionamiento o división
5. Inversión
6. Brainstorming
7. Analogías

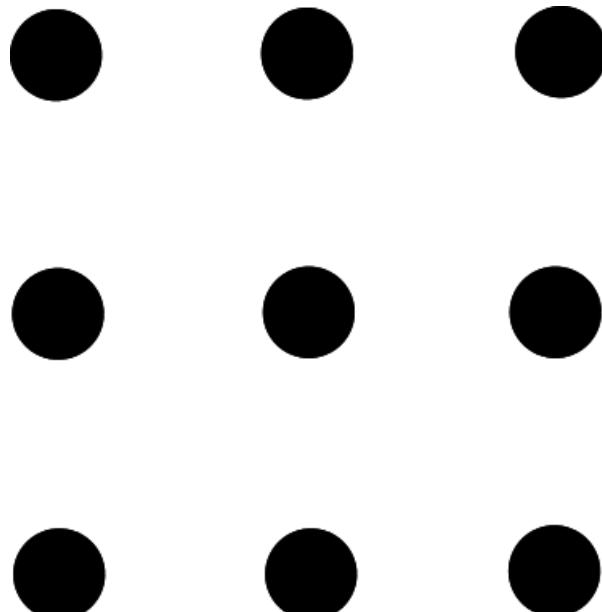
Técnica 1: Revisar Supuestos

La primer técnica que vamos a ver es la **revisión de supuestos**. Es simple y está relacionado a la manera en la que axiomatizamos los conceptos y las ideas. Dar algo por supuesto (supuestos lógicos que se aceptan por válidos en sí mismos), nos ayuda a construir un modelo sobre esto sin preocuparnos, muchas veces, por las cuestiones triviales. Además, la aceptación de que una idea sea correcta, no garantiza su corrección.

Práctica para el dojo:

Ejercicio: El siguiente problema ilustra muy bien el factor restrictivo de los supuestos. Consiste en unir los nueve puntos mediante sólo 4 líneas rectas pero sin levantar el lápiz del papel.

⁸⁴<http://germanbraun.blogspot.com.ar/2013/02/habilidades-de-un-tester-el-quid-de-la.html>



Testing Lateral

Ayuda: El factor que bloquea la solución es que las líneas rectas han de unir los puntos sin exceder los límites de los propios puntos. Los invito a que intenten resolver el ejercicio.

En testing, hacer muchas suposiciones puede ser peligroso. Estas pueden estar asociadas, por ejemplo, a la estandarización de ciertos sistemas. *La interface de una aplicación es la misma para todas las ventanas, funciona de la misma manera que el resto y, por ende, lo hace bien.* En conclusión, la revisión intenta demostrar que cualquier supuesto puede ser sometido a examen. Claramente, esta técnica no intenta revisar todos los supuestos posibles. No sería aplicable y rentable para ningún proyecto. Sin embargo, para un tester, tener este pensamiento crítico es fundamental. Como ya dijo Jerry Weinberg: “*A tester is someone who knows that things could be different*”⁸⁵.

Hasta dónde hemos llegado...

Hace unas semanas, una persona me contactó por LinkedIn⁸⁶ para que le comentase sobre testing (en general). Aquí dejo algunas reflexiones de mi respuesta.

Recuerdo que +Ernesto Kiszkurno⁸⁷ (socio de la consultora en la cual trabajo), siempre me decía que cuando empezaron a trabajar en testing, a ofrecer y vender servicios, los miraban como “personas extrañas”. Nadie entendía lo que querían hacer y, obviamente, había una resistencia importante. Esto hace ya unos 15 años. Esta resistencia, creo yo, era normal. Alguien te venía a decir que tu software “estaba mal”. ¿Cómo es esto?

⁸⁵<http://germanbraun.blogspot.com.ar/2013/02/mi-responsabilidad-como-tester.html>

⁸⁶<http://www.linkedin.com/>

⁸⁷<http://plus.google.com/110288085067434958046>

Hoy en día, el conjunto de tareas y prácticas que involucran al testing ya está totalmente aceptado y es indispensable para el desarrollo de todo producto software. Como consecuencia, creo que la responsabilidad nuestra (como testers) también se ha potenciado. Los testers debemos dar soporte a la [captura del conocimiento](#)⁸⁸, trabajamos con conocimiento y eso también hay que gestionarlo. No sólo debemos detectar ciertos desvíos funcionales o errores triviales que pueden tener cualquier producto, sino hay que involucrarse mucho más en el dominio. Las prácticas ágiles (TDD, ATDD, BDD, DDD) que involucran desarrollo basado en pruebas, son también un punto de encuentro entre developers y testers que potencia su relación y realza la necesidad de trabajar con foco en la calidad. Finalmente, creo que el próximo paso es aceptar y dejar de lado el famoso “[developers vs testers](#)”⁸⁹ para pasar a ser un grupo homogéneo de profesionales trabajando en pos de entregar valor. Un equipo multidisciplinario que tiene como único objetivo, construir software de calidad. La gente depende cada vez más del software y nosotros debemos dar soporte a esto.

En resumen, enumero los siguientes hitos que se fueron sucediendo en el tiempo y, que espero, puedan terminar de concretarse en un futuro no muy lejano.

1. Resistencia al testing.
2. Aceptación como la última etapa de un proceso de desarrollo cascada (waterfall)
3. Necesidad de incluir prácticas de calidad más temprano en el ciclo.
4. Surgimiento de testing agile (TDD, BDD, ATDD) para trabajar sobre el desarrollo gestionado con pruebas.
5. Testing como actividad que retroalimenta la captura del conocimiento del software.
6. Fin de la rivalidad “[developers vs testers](#)”
7. Imposición de un nuevo concepto **QDD (Quality-Driven Development)**

⁸⁸<http://germanbraun.blogspot.com.ar/2013/07/hacer-testing-es-tambien-capturar.html>

⁸⁹<http://germanbraun.blogspot.com.ar/2013/09/basta-de-dev-vs-qa.html>

Agilidad

Agile is in the air

El 11 de Mayo de 2013 participé en el segundo encuentro del “Agile is in the air”. La filosofía de estos encuentros es, básicamente, hablar de agilidad al aire libre. En el caso de ayer, la temática era “Juegos para transmitir la agilidad”.

Comenzó con un Open Space para definir los temas a tratar. De aquí salieron 3 *tracks con es el siguiente orden luego de una votación tipo twister*: “Scrum con el cuerpo”, “Gamification de Scrum” y “Juegos para retrospectivas”. Sólo se trataron las dos primeras.

Les dejo el video del primer *track* “Scrum con el cuerpo”⁹⁰. La idea de esta sesión era representar los roles, artefactos y encuentros de Scrum usando el cuerpo y la voz de manera que sea más fácil transmitir y recordar las reglas del framework.

Finalmente, durante el segundo *track* se delineó una aplicación para gamificar equipos ágiles.

Estuvo muy bueno. Espero poder participar del próximo.

Curso CSM

Los días Lunes 10 y Martes 11 de Junio de 2013 hice el curso para la certificación Scrum Master (CSM). ¡Estuvo excelente! Muchas cosas para masticar...

Muchas gracias Alan Cyment, Diego Fontdevila y Verónica Sack. Aquí algunas fotos:

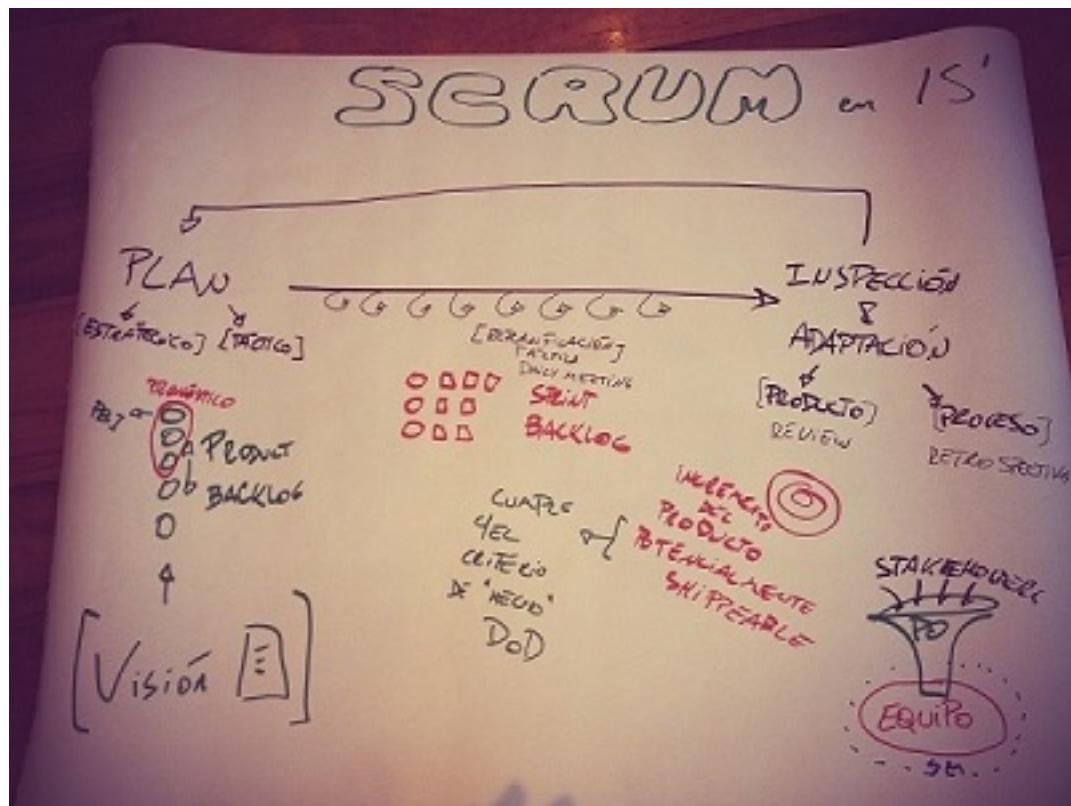


csm 1

⁹⁰https://www.youtube.com/watch?feature=player_embedded&v=3y2MKfrInmU



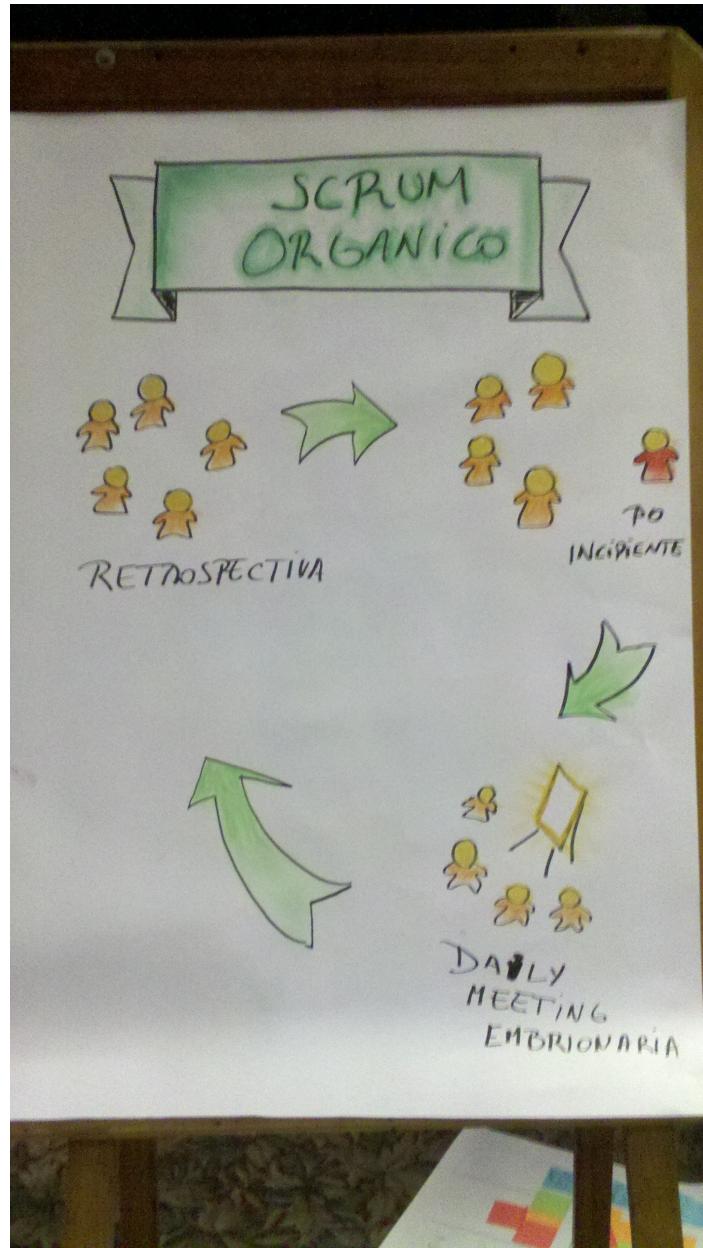
csm 2



csm 3



csm 4

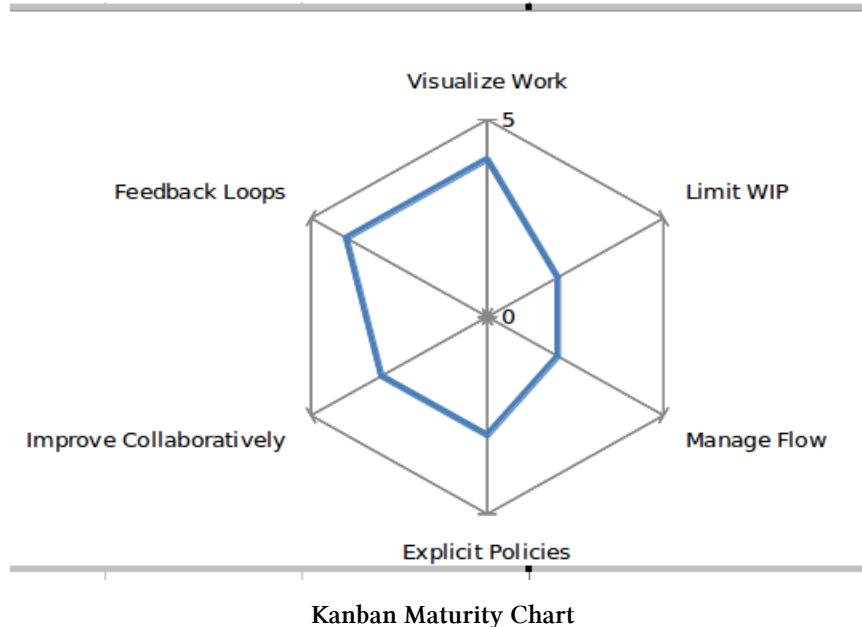


csm 5

Kanban Maturity Chart

Hace un tiempo encontré este [Kanban Maturity Chart⁹¹](#) que me pareció muy piola para mediar a nuestro equipo y usarlo en retrospectivas.

⁹¹<http://paulklipp.com/images/KanbanMaturityChart.pdf>



Visualize Work: How well does the team consistently use visualization tools to make the flow of work through the system visible?

Limit WIP: Does the team set WIP limits and use them as a tool for managing flow? Do they review WIP limits, adjust them, and respect them (which is not the same as blindly obeying them)

Manage Flow: How well does the team consistently observe the flow of work using relevant measures as guides in a continuing process of reducing waste, variability, and improving the flow of work through the system?

Explicit Policies: Does the team make policies relevant to the handling of tasks clear and explicit and review those policies as needed?

Improve Collaboratively: How well does the team use models and experiments to test ideas in order to introduce improvements to the process?

Feedback Loops: How well does the team create feedback loops to give both the team and individuals actionable feedback on quality and process?

¿Ya lo probaron? ¿Lo conocían? Espero los comentarios.

Mapas de Impacto

Estoy leyendo el libro de [Gojko Adzic⁹²](#), “Impact Mapping: Making a big impact with software product and projects”⁹³. Me tiene bastante entretenido. Es un tema que está haciendo mucho ruido.

⁹²<https://twitter.com/gojkoadzic>

⁹³<http://www.impactmapping.org/book.php>

Personalmente, estoy intentando trabajar con mapas mentales desde hace un tiempito. Creo que es una herramienta potente que puede aplicarse a diversos dominios, tareas y/o problemas que tengamos que resolver. Sobre una aplicación puntual ya he escrito [aqui⁹⁴](#).

Un mapa de impacto es un mapa mental que permite a los equipos y a las organizaciones visualizar alcance y asunciones relacionadas a algún objetivo que se intente lograr. Supongamos que, en nuestro contexto, es construir un producto software. Estos mapas son creados de manera colaborativa por todos los involucrados e interesados en lograr la **meta **que los convoca (obviamente, esto es una situación ideal). La gran ventaja de estos mapas: permiten tener controlada la *big picture* de nuestro proyecto, tener en claro a dónde queremos ir, cómo llegar y a quiénes debemos involucrar.

A modo de ilustración, hice el siguiente mapa de impacto usando la herramienta [MindMup⁹⁵](#).



Tener +500 visitas en una entrada sobre Mapas de Impacto

Durante la creación de una mapa es necesario responder a las siguientes preguntas:

- **WHY?** Es el centro del mapa. Es la **meta** que se está tratando de lograr. Una buena meta debería ser SMART (Specific, Measurable, Action-oriented, Realistic and Timely) y presentar el problema ha ser resuelto.
- **WHO?** Este nivel indica quienes son los **actores** que pueden influenciar el resultado. Quién puede producir el efecto deseado, quién puede obstruirlo y quién será impactado. Debe ser lo más específico posible.
- **HOW?** Explicita el **impacto **que se trata de crear. Cómo debe cambiar la conducta de los actores y cómo pueden ayudar a lograrlo. Especificar estos impactos permite priorizar e identificar riesgos.
- **WHAT?** El último nivel del mapa es el nivel de los **entregables**. Qué puede hacer el equipo o la organización para dar soporte al impacto. Puede contener varios niveles más y no necesariamente deben completarse desde el inicio. Puede ser refinado de manera iterativa.

Espero que se animen a aplicarlo y a compartir la experiencia :) Voy a seguir escribiendo sobre el tema a medida que avance con el libro.

⁹⁴<http://germanbraun.blogspot.com.ar/2013/03/mind-mapping.html>

⁹⁵<http://www.mindmup.com/>

We don't know exactly where we'll go but small steps ensure we do not fall down

Ver el todo de un proyecto es algo que puede parecer simple, pero que necesita de sus herramientas y recursos para poder hacerlo de manera controlada y efectiva.

Hasta aquí, hemos estado hablando mucho sobre agilidad y sus bondades. Sin embargo, aparece el siguiente efecto colateral: perdemos de vista “el todo” del proyecto. Es decir, ¿estamos haciendo lo que realmente debemos hacer?, ¿hacia dónde vamos? ¿dónde estamos parados actualmente luego de finalizar determinadas tareas? Si lo trasladamos a un proyecto de testing, ¿todas las funcionalidades están cubiertas por nuestras pruebas? ¿dónde nos está faltando testing? ¿cómo está el semáforo del proyecto?

Gojko Adzic⁹⁶, en su libro “Impact Mapping: Making a big impact with software product and projects”⁹⁷, se encarga de tratar este tema usando los mapas mentales. Adzic, nota que una causa común de la desconexión entre el negocio y el *delivery* es que los equipos que trabajan en un esquema iterativo, lo hacen sobre pequeñas funcionalidades que tienden a ir por un *pipeline* sin tener en cuenta su valor para el negocio. Con el foco puesto justamente en el *delivery*, los **mapas de impacto** facilitan esta vista tanto para el negocio como para el equipo técnico y se adaptan bien a los modelos iterativos. Usando mapas de impacto podemos reportar progreso, comprometernos sobre impactos a ser logrados, dar visibilidad y permitir una implementación más flexible.

Para Janet Gregory⁹⁸ y Lisa Crispin⁹⁹, *look at the big picture* es uno de los factores claves de éxito según definen en su libro “Agile Testing: A Practical Guide for Testers and Agile Teams”¹⁰⁰. Para ellas, todos los *software makers*¹⁰¹ deben mirar el todo del proyecto y usualmente desde el punto de vista del cliente. ¿Qué sucede si una nueva funcionalidad causa que otras no relacionadas rompan? Es necesario considerar el impacto sobre el sistema completo y llamar la atención del equipo sobre esto. Enfocándose puntualmente en testing, Crispin y Gregory proponen evaluar cómo las actuales *stories* se insertan en el esquema del negocio y preguntarnos a nosotros mismos cómo podemos hacer un mejor trabajo para poder entregar valor real. Usar el **cuadrante de testing ágil**¹⁰², guiar el desarrollo con tests, hacer testing exploratorio para aprender sobre el negocio, hacer un ambiente de pruebas tan similar al productivo como sea posible, usando datos reales y recrear situaciones en producción, tales como el testing de carga, son algunas de las herramientas que se proponen en el libro.

Personalmente, he puesto en práctica varias de estas herramientas y consejos en mis proyectos. Algunas ya las hemos charlado y otras las vamos a ir desarrollando posteriormente. Sin embargo, antes de seguir avanzando, creo que es necesario ponernos a pensar si estamos viendo el todo de

⁹⁶<https://twitter.com/gojkoadzic>

⁹⁷<http://www.impactmapping.org/book.php>

⁹⁸<http://ca.linkedin.com/in/janetgregory>

⁹⁹<http://www.linkedin.com/pub/lisa-crispin/0/20a/884/>

¹⁰⁰<http://www.amazon.com/Agile-Testing-Practical-Guide-Testers/dp/0321534468>

¹⁰¹<http://germanbraun.blogspot.com.ar/2013/09/basta-de-dev-vs-qa.html>

¹⁰²<http://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/>

nuestros proyectos.

¿Ustedes lo están haciendo?

[1] El título de la nota fue extraído del libro “Impact Mapping: Making a big impact with software product and projects”¹⁰³

Así comienza el camino...

”Scrum: learn from a new start. Kanban: learn from where we are now”.

Si queremos transicionar de una esquema “más Waterfall” a un entorno ágil y no estamos del todo preparados, es recomendable empezar por Kanban. Si bien ya he hablado de esta herramienta en varias ocasiones, me parece importante definir qué debemos tener en cuenta a la hora de aventurarnos en la agilidad desde cero.

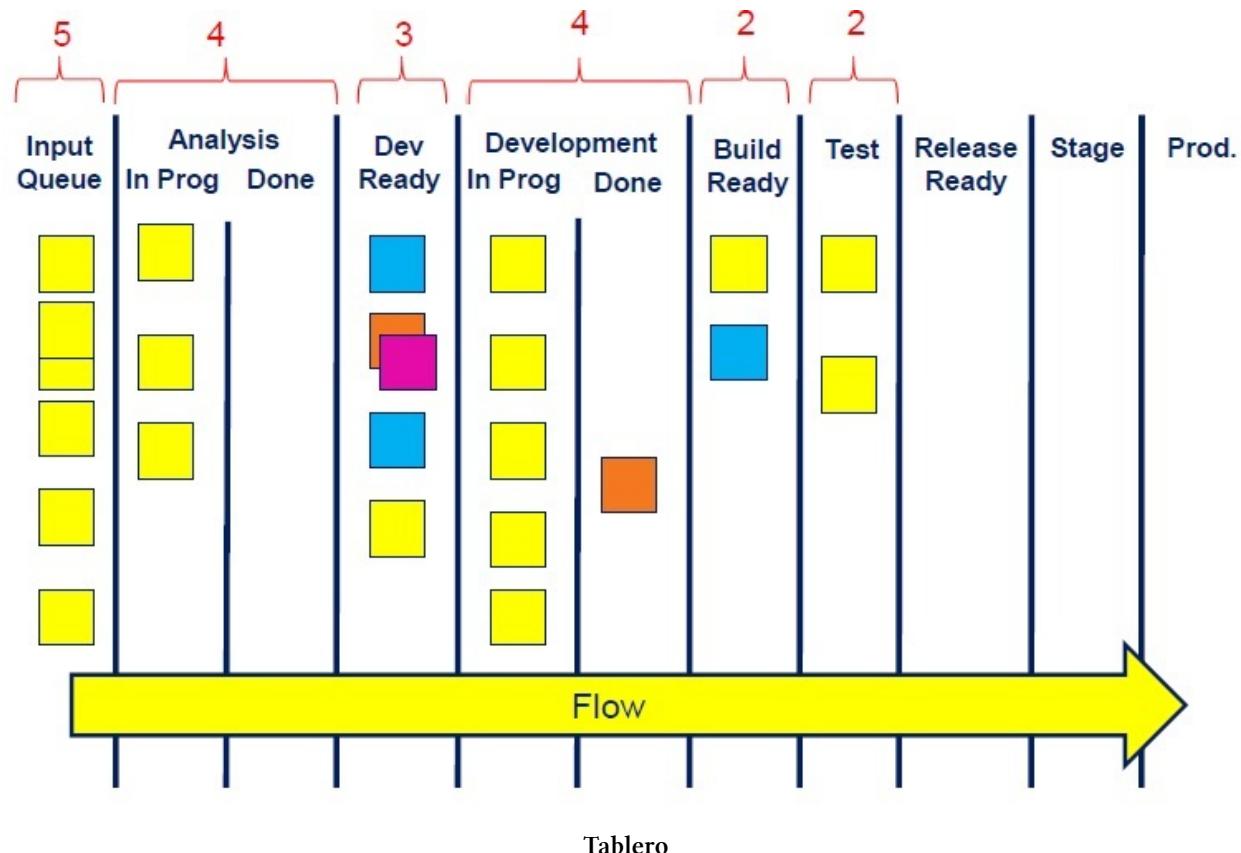
¿Se puede combinar Kanban con mi proceso actual?

Si, y podemos empezar de la siguiente manera.

- **Tablero:** comenzamos visualizando nuestro trabajo. ¿Dónde están nuestros cuellos de botella?
- **WIP (Work in Progress):** ¿cuánto trabajo podemos tomar en cada etapa de nuestro proceso?

Así comienza el camino...

¹⁰³<http://www.impactmapping.org/book.php>



[Entradas relacionadas]

- Una experiencia aplicando Kanban en Testing¹⁰⁴
- Lean¹⁰⁵
- Kanban Maturity Chart¹⁰⁶

[Referencias]

- <http://www.crisp.se/gratis-material-och-guider/kanban>¹⁰⁷
- Kanban: Successful Evolutionary Change for Your Technology Business¹⁰⁸

¹⁰⁴<http://germanbraun.blogspot.com.ar/2013/05/una-experiencia-aplicando-kanban-en.html>

¹⁰⁵<http://germanbraun.blogspot.com.ar/2013/11/lean.html>

¹⁰⁶<http://germanbraun.blogspot.com.ar/2013/12/kanban-maturity-chart.html>

¹⁰⁷<http://www.crisp.se/gratis-material-och-guider/kanban>

¹⁰⁸<http://www.amazon.com/Kanban-Successful-Evolutionary-Technology-Business/dp/0984521402>

¿Para qué Scrum?

Basándome en este [artículo¹⁰⁹](#), ¿Qué es Scrum para mí? escrito por [Martín Alaimo¹¹⁰](#), me propuse hacer uno propio, con algunas reflexiones. Hablar de Scrum, por mencionar un *framework* conocido y particular, pero en realidad, sobre una cultura de trabajo colaborativa.

[Xavier Quesada Allue¹¹¹](#) dice que Scrum sirve para ayudarme a aumentar Productividad, Calidad y Felicidad. Xavier pone el foco inicial en la Calidad. Entonces dice que logrando Calidad, vamos a lograr Productividad y así la Felicidad. Por su parte, [Alan Cyment¹¹²](#), promueve una [parecida¹¹³](#). Sin embargo, el foco es la Felicidad. La Felicidad genera el espacio para la Productividad y Calidad.

Ambos enfoques son interesantes. Sin embargo, a esto le quiero agregar un concepto que tiene que ver con lo que para mi es Scrum, o para qué sirve. Cuando hice el CSM (justamente con Cyment, en Junio de 2013), la última consigna del curso era que, antes de despedirnos, cada uno diga, qué aprendieron en el curso, qué palabra, qué frase nos llevamos de la capacitación. En ese momento dije: *crecimiento orgánico*. Hoy creo que este es el concepto más fuerte de Scrum y de cualquier entorno colaborativo e iterativo. Manejar la complejidad, creciendo orgánicamente para que “mi planta no se pudra”.

¡Necesitamos tener un tablero y una metodología, como sea!

A menudo sucede que me encuentro hablando con gente que piensa que hacer algo *agile* es implementar un tablero y con eso ya está (me ha pasado mucho en los últimos 3 o 4 meses). Verdaderamente, están haciendo algo: visualizando su trabajo en curso, sus lista de tareas y sus desvíos. Lo cual no es poco, pero...

El furor por las metodologías ágiles es tal que por momentos algunas organizaciones necesitan aplicarlas a cualquier precio, cómo sea. Quizás sea un problema de “vacío metodológico” en dichas organizaciones, potenciado por el hecho de que algunos *frameworks* tienen pocas reglas, son “relativamente simple” y el tiempo de aplicación es corto. También, muchas veces existe una especie de “bajada de línea” de las gerencias sin permitir a sus equipos madurar y hacer crecer orgánicamente su necesidad de trabajar en un esquema colaborativo, iterativo y con todo lo que eso significa.

En conclusión, las metodologías ágiles son fáciles de aprender (son *frameworks* con pocas reglas) pero difíciles de aplicar correctamente. Sin embargo, pienso que también se han vuelto populares debido a que las organizaciones (no todas, pero bastantes) tienen este “vacío” que muchas veces atenta contra su buen uso.

¹⁰⁹<http://www.martinalaimo.com/blog/que-es-scrum-para-mi>

¹¹⁰https://www.google.com.ar/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCkQFjAA&url=http%3A%2Fwww.martinalaimo.com%2F&ei=5dXTUonEFcivsASp3YHwCw&usg=AFQjCNHoVBysNNNl4s5mCoq6D_jqDSR62g&sig2=PAuBiH7K6wd4T9QEtlS6rw&bvm=bv.59026428,d.cWe

¹¹¹<http://www.xqa.com.ar/visualmanagement/>

¹¹²<http://www.cyment.com/>

¹¹³<http://germanbraun.blogspot.com.ar/2013/03/responsabilidad-cambios-y-motivacion-mi.html>

Entrevistas y Aportes

Durante el 2013 inauguré un ciclo de entrevistas (que pienso continuar) en las cuáles tuve la posibilidad de charlar con gente de mucha experiencia en software, testing y metodologías ágiles. Fue una experiencia muy enriquecedora y muy bien recibida por los lectores. Aquí las dejo para que las disfruten igual que yo.

Entrada en el blog de PetroVR

Colaboré como autor, junto a mis compañeros de equipo de [Pragma¹¹⁴](#), en una entrada sobre Testing en el blog de [PetroVR¹¹⁵](#). En la entrada, hablamos un poco sobre una propiedad que me parece interesante para desarrollar y que es la “self-testability” de un producto software. Los invito a leerlo [How is the Testing of PetroVR done? The self-testability of PetroVR¹¹⁶](#)

Gracias [Caesar Systems¹¹⁷](#) por la invitación a participar y a [+Leandro Caniglia¹¹⁸](#) y [+Carlos Ferro¹¹⁹](#) por el feedback.

Sobre Pragma Consultores

Pragma Consultores es un grupo multinacional con impronta en Latinoamérica y España. Pragma provee servicios de Consultoría, Calidad, Gestión de Proyectos y Tecnología a las compañías más importantes dentro de una gran variedad de industrias. Pragma combina diversos conocimientos, culturas y experiencias. Sus profesionales comparten metodologías, estándares operativos y técnicos así como capacitaciones y tecnologías.

Sobre Caesar Systems

Caesar Systems ayuda a las compañías E&P a acelerar su producción a través de su suite PetroVR, dando soporte global en materia de coaching. El equipo Caesar Systems está altamente calificado para liderar equipos de proyectos en compañías y guiarlos en la toma de decisiones para cumplir con los desafíos del negocio.

Sobre PetroVR

Con más de 15 años, PetroVR es un software estándar global para asegurar la toma de decisiones en la industria de E&P. PetroVR simplifica la planificación y el desarrollo de un activo de negocio para la toma de decisiones en la industria de la producción y explotación de petróleo.

¹¹⁴<http://pragmaconsultores.com/Paginas/Home.aspx>

¹¹⁵<http://www.caesarsystems.com/petrovr/>

¹¹⁶<http://www.caesarsystems.com/blog/testing-petrovr-done-self-testability-petrovr/#.UtaI7dLuJdM>

¹¹⁷<http://www.caesarsystems.com/>

¹¹⁸<http://plus.google.com/113622131200072499744>

¹¹⁹<http://plus.google.com/108688683714102694833>

Testing: 6 preguntas a Ernesto Kiszkurno de Pragma Consultores

Siguiendo con mi serie de entrevistas, estuve charlando con Ernesto Kizskurno¹²⁰, de Pragma Consultores¹²¹, sobre testing de software.

German Braun (GB): ¿Cuál es su definición de calidad?

Ernesto Kiszkurno (EK): *Tengo 10 definiciones. Pero la que más me gusta es: “La calidad no es un arte, es un hábito” de Aristóteles. El resto las pueden encontrar en mi blog*¹²²

GB: ¿Cuál piensa que es, actualmente, el rol del testing en un proceso de desarrollo u organización?

EK: *Hace unos días publiqué en twitter*¹²³ *algo relacionado con esto. El testing no tiene valor en si mismo para una organización. No es ni bueno ni malo en sí mismo. Es una herramienta para medir el nivel de calidad del software que desarrollamos. Eso no quiere decir que no sea fundamental su uso en un montón de ámbitos. Lo que me parece clave es entender el contexto del proceso de desarrollo de software y en base a eso decidir cuánto invertimos en testing.*

*Dicho esto y, dado el nivel de penetración que tiene el software en nuestra vida cotidiana, creo que tendremos testing para rato. Sobre esto también comenté algo aquí*¹²⁴.

GB: ¿Cuáles piensa que son las principales skills que un tester debería tener?

EK:

1. *Meticulosidad.*
2. *Curiosidad para buscar los errores, para entender la funcionalidad.*
3. *Buenas habilidades de comunicación (escrita y oral).*
4. *Solvencia técnica (para poder hablar con el desarrollador, entre otras cosas).*
5. *Entendimiento del negocio (para poder hablar con los usuarios).*
6. *Mindset*¹²⁵ *diferente.*

GB: ¿Es el testing una “profesión” con futuro?

EK: *He escrito sobre esto aquí*¹²⁶. Creo que el testing como actividad debe seguir re-inventándose a medida que la práctica de desarrollo de software vaya cambiando. Pero no me imagino un futuro inmediato donde hay desarrollo de software pero no hay testing.

Por otro lado, el paradigma de comportamientos estancos entre el grupo de desarrollo de software y el equipo de testing se va agotando.

¹²⁰<https://twitter.com/ekiszkurno>

¹²¹<http://pragmaconsultores.com/Paginas/Home.aspx>

¹²²<http://ernestokiszkurno.blogspot.com.ar/2012/05/10-definiciones-de-calidad.html>

¹²³<http://twitter.com/ekiszkurno/statuses/291721985862537216>

¹²⁴<http://ernestokiszkurno.blogspot.com.ar/2011/02/si-no-tengo-errores-en-produccion-por.html>

¹²⁵<http://ernestokiszkurno.blogspot.com.ar/2010/02/tester-mindset.html>

¹²⁶<http://ernestokiszkurno.blogspot.com.ar/2012/06/el-futuro-del-testing.html>

GB: ¿Cuáles son los 2 o 3 hitos más importantes de su carrera como tester profesional?

EK: *Es difícil esta pregunta porque en la época donde yo empecé con los temas de calidad eramos pocos. Creo que los hitos más importantes en mi carrera fueron los momentos donde dí saltos de magnitud en el tipo de responsabilidades que asumía. Por ejemplo el cambio de ser liderado a liderar o el cambio de liderar un pequeño equipo a muchos equipos.*

GB: ¿Qué consejo le daría a un tester principiante que desea seguir el camino de la calidad de software?

EK: *Que cuide su empleabilidad¹²⁷, aprenda continuamente y que no se concentre sólo en los aspectos técnicos. Lo más complicado es el manejo de las cuestiones soft.*

Sobre Ernesto Kiszkurno

Ernesto es socio en [Pragma Consultores¹²⁸](#). También es docente en la [Facultad de Ciencias Exactas y Naturales¹²⁹](#) de la [UBA¹³⁰](#) y autor del blog “[Así no se hacen las cosas](#)¹³¹”.

¡Gracias Ernesto!

Desarrollar Software: 6 preguntas a Leandro Caniglia de Caesar Systems

Inicié una serie de entrevistas a diferentes personas con las que tengo el gusto de trabajar y que me han enseñado mucho. En este caso, entrevisté a Leandro Caniglia de [Caesar Systems¹³²](#), con quién he trabajado en los últimos años como tester de [PetroVR¹³³](#).

Germán Braun (GB): ¿Puede describir, en pocas palabras, qué es desarrollar software?

Leandro Caniglia (LC): *Desarrollar software es capturar conocimiento. Y como ese conocimiento se capture siguiendo una metáfora uniforme (la de tu software), la captura dà rápidamente paso a la creación de nuevo conocimiento.*

GB: El desarrollo de software es en sí, un proceso complejo. Sin embargo, ¿dónde radica la mayor complejidad?

LC: *La complejidad del software radica en el conflicto que hay entre la tolerancia a fallas que caracteriza a los seres biológicos y la exacerbante precisión de las computadoras, que tiene que ser estricta. A partir de ahí la complejidad suele crecer cuando los programadores escriben código robusto para defenderse de posibles errores.*

¹²⁷<http://ernestokiszkurno.blogspot.com.ar/2012/05/empleabilidad.html>

¹²⁸<http://pragmaconsultores.com/Paginas/Home.aspx>

¹²⁹<http://exactas.uba.ar/>

¹³⁰<http://www.uba.ar/>

¹³¹<http://ernestokiszkurno.blogspot.com.ar/>

¹³²<http://www.caesarsystems.com/>

¹³³<http://www.caesarsystems.com/petrovr/>

GB: ¿Podría enunciar algunos tips que se deberían tener en cuenta para desarrollar software de calidad?

LC: *Las reglas son pocas y sencillas. La dificultad estriba en aplicarlas todas todo el tiempo. Las más importantes son:*

i) La partición del trabajo en tareas cortas y bien definidas cuya realización no lleve mas que unas pocas decenas de líneas de código.

ii) La revisión activa de cada change set. En esta práctica el revisor no le sugiere al autor cómo mejorar el código, lo modifica y produce mejor código siempre que pueda y sin importar si el cambio es insignificante o profundo.

iii) Todo change set debe tener al menos un test que ejercite la parte relevante de la tarea. Los tests unitarios debe correr en menos de 100 ms (cada uno).

iv) Integración continua, previo a correr todos los tests unitarios.

v) Testing manual a cargo de un equipo de testers profesionales independiente de los programadores.

vi) Refactoring permanente. Todos tocan todo, incluso lo que anda.

GB: Existen diferentes lenguajes de programación que implementan diferentes paradigmas. ¿Cuál es el lenguaje de programación que utiliza? ¿Por qué?

LC: *Smalltalk. Lo utilizo porque es el mejor y en el mercado en el que compite Caesar el que dá ventaja pierde.*

GB: ¿Cuál es su opinión sobre el uso de metodología ágiles?

LC: *Opino que las metodologías ágiles han iluminado el panorama del desarrollo de software y lo han enriquecido. Hoy en día los buenos equipos de desarrolladores están mejor preparados para gerenciar cualquier tipo de proyecto (de software o de la industria que fuere) que sus clientes más encumbrados.*

GB: ¿Es necesario el testing de software? ¿Aporta valor al producto final?

LC: *El testing de software es esencial. Todo aquel que quiera producir software debería tener un equipo de testers profesionales de tamaño comparable al de programadores. Para desarrollar software se puede prescindir de todo el mundo, menos de los programadores y los testers.*

Sobre Leandro Caniglia

Leandro es *chief technologist* en [Caesar Systems¹³⁴](http://www.caesarsystems.com/). También es fundador y presidente de la [FAST¹³⁵](http://www.fast.org.ar/), la Fundación Argentina de Smalltalk.

¡Gracias Leandro!

¹³⁴<http://www.caesarsystems.com/>

¹³⁵<http://www.fast.org.ar/>

Agile: 6 preguntas a Thomas Wallet de Pragma Consultores

Último post de la serie de entrevistas que comenzó con [Leandro Caniglia, sobre desarrollo de software¹³⁶](#), y siguió con [Ernesto Kiszkurno, sobre testing¹³⁷](#). Esta vez estuve charlando con [Thomas Wallet¹³⁸](#), de [Pragma Consultores¹³⁹](#), sobre “el mundo de la agilidad”.

German Braun (GB): ¿Podría decirme, en pocas palabras, qué es “ser ágil”?

Thomas Wallet (TW): *Para mi, ágil es una búsqueda, más que un estado. “Ser ágil” es estar buscando constantemente la mejora del trabajo en equipo, cuidando particularmente el compromiso con la transparencia, la auto-gestión, el desarrollo de las personas, la entrega frecuente y el feedback constante.*

GB: ¿Por qué usa y pregoná el uso de metodologías ágiles?

TW: *Primero, estoy personalmente muy alineado y convencido de los valores que suelen sostener las prácticas ágiles que conozco. Y segundo, porque estoy íntimamente convencido que muchas empresas funcionan mal y/o no permiten un desarrollo sano de las personas, problema al cual, la implementación de ciertas prácticas y valores ágiles, puede dar buenas respuestas.*

GB: ¿Cuáles son los mayores desafíos a la hora de implementar una metodología ágil?

TW: *La resistencia al cambio de paradigma es una característica humana que suele ser la principal dificultad de las implementaciones de prácticas ágiles que he vivido. En particular, la cultura organizacional que suelen tener muchas empresas choca con los principios de transparencia y auto-gestión.*

GB: ¿En qué ambientes considera que son más exitosas estas metodologías?

TW: *No veo límites para la aplicación de las prácticas ágiles Si bien son más usadas y conocidas en el desarrollo de software, se extienden cada vez más en otros dominios. Suelen ser exitosas en ambientes dinámicos con mucha presión sobre el time-to market (startups, proyectos de desarrollo en crisis, etc.) Si nos referimos al modelo de cultura organizacional de William Schneider, suele ser más sencilla la implementación de prácticas ágiles en organizaciones de Colaboración y/o Crecimiento que en organizaciones de Control y/o Capacidad.*

GB: ¿Cuál es el futuro del “movimiento ágil”?

TW: *Creo que vamos a vivir en la próxima década una explosión de la aplicación del agilismo en otras especialidades que el desarrollo de software, al mismo tiempo que las prácticas van a ser cada vez más mainstream en el desarrollo.*

Creo que es necesario que se re-inventen nuevas metodologías ágiles, ya que las principales (Scrum, XP) rondan los 20 años de existencia. Me atrevo a predecir el futuro nacimiento de metodologías

¹³⁶<http://germanbraun.blogspot.com.ar/2013/01/desarrollar-software-6-preguntas.html>

¹³⁷<http://germanbraun.blogspot.com.ar/2013/02/testing-6-preguntas-ernesto-kiszkurno.html>

¹³⁸<https://twitter.com/WalletThomas>

¹³⁹<http://pragmaconsultores.com/Paginas/Home.aspx>

ágiles minimalistas (pocas reglas), centradas en la gestión de MMFs (Minimum Marketable Feature) y en la mejora continua.

GB: Por último, ¿podría recomendarnos algunos blogs, podcast o libros para lectores interesados?

TW: Últimamente me interesan mucho los blogs de Tobias Mayer (<http://businesscraftsmanship.com/>¹⁴⁰), John Somez (<http://simpleprogrammer.com/>¹⁴¹) e Yves Hanouille (<http://www.hanouille.be/>¹⁴²).

Y dos libros que fueron mi puerta de entrada al mundo de la agilidad (aunque no sean reconocidos como libros ágiles):

- *Peopleware: Productive Projects and Teams*, de Tom DeMarco y Timothy Lister
- *Project Retrospectives: A Handbook for Team Reviews*, de Norman Kerth

Sobre Thomas Wallet

Thomas es consultor PMO en el Banco Provincia del Neuquén¹⁴³. Thomas cuenta con amplia experiencia en aplicación de metodologías ágiles en diferentes proyectos de variados mercados. Además, es autor del blog “el próximo paso¹⁴⁴”.

¡Gracias Thomas!

¹⁴⁰<http://businesscraftsmanship.com/>

¹⁴¹<http://simpleprogrammer.com/>

¹⁴²<http://www.hanouille.be/>

¹⁴³<http://pragmiconsultores.com/Paginas/Home.aspx>

¹⁴⁴<http://thomaswallet.blogspot.com.ar/>

Creatividad y Pensamiento

How do I find the next idea?

Ideas on how to make new questions... Muy buen keynote de Leandro Caniglia en la Smalltalks 2013.

Video - How do I find the next idea?¹⁴⁵

Mantenerse creativo

Hace un tiempo encontré una lista de [33 tips para mantenerse creativo¹⁴⁶](#). Me pareció interesante, y por supuesto, me propuse seguirlos (por ahora parcialmente). Voy a comentarles cuales fueron mis elecciones para cada uno de los tips que pude implementar, por lo tanto, a partir de aquí, comienza una saga de posts con mis recomendaciones. Ustedes, ¿cómo se mantienen creativos?

Si les interesan los tips (en general), también estoy siguiendo los [_consultips _de “Así no se hacen las cosas”¹⁴⁷](#) y los [_retrotips _de “el próximo paso”¹⁴⁸](#). Muy interesantes para aplicar.

Comencemos...

#1 “Haga listas”

Tan simple como eso. Pueden ser en papel o cualquier dispositivo digital más avanzado. No importa, hay que sacarse las cosas de la cabeza y explicitarlas de tal manera que podamos atenderlas una por una a su debido tiempo. Podemos hacer listas de nuestros proyectos, tareas de la semana, ideas...

Las listas, por ejemplo, son un pilar fundamental de cualquier implementación [GTD \(“Getting Things Done”\)¹⁴⁹](#) propuesta por [David Allen¹⁵⁰](#). Actualmente, estoy haciendo mi propia implementación GTD pero aún le falta madurar. Cuando esté aceptada, se las comparto.

Mantenerse creativo (2)

Seguimos con [33 tips para mantenerse creativo¹⁵¹](#).

¹⁴⁵https://www.youtube.com/watch?feature=player_embedded&v=tuOqJU9BtkQ

¹⁴⁶<http://www.yorokobu.es/33-formas-para-seguir-siendo-creativo/>

¹⁴⁷<http://ernestokiszkurno.blogspot.com.ar/>

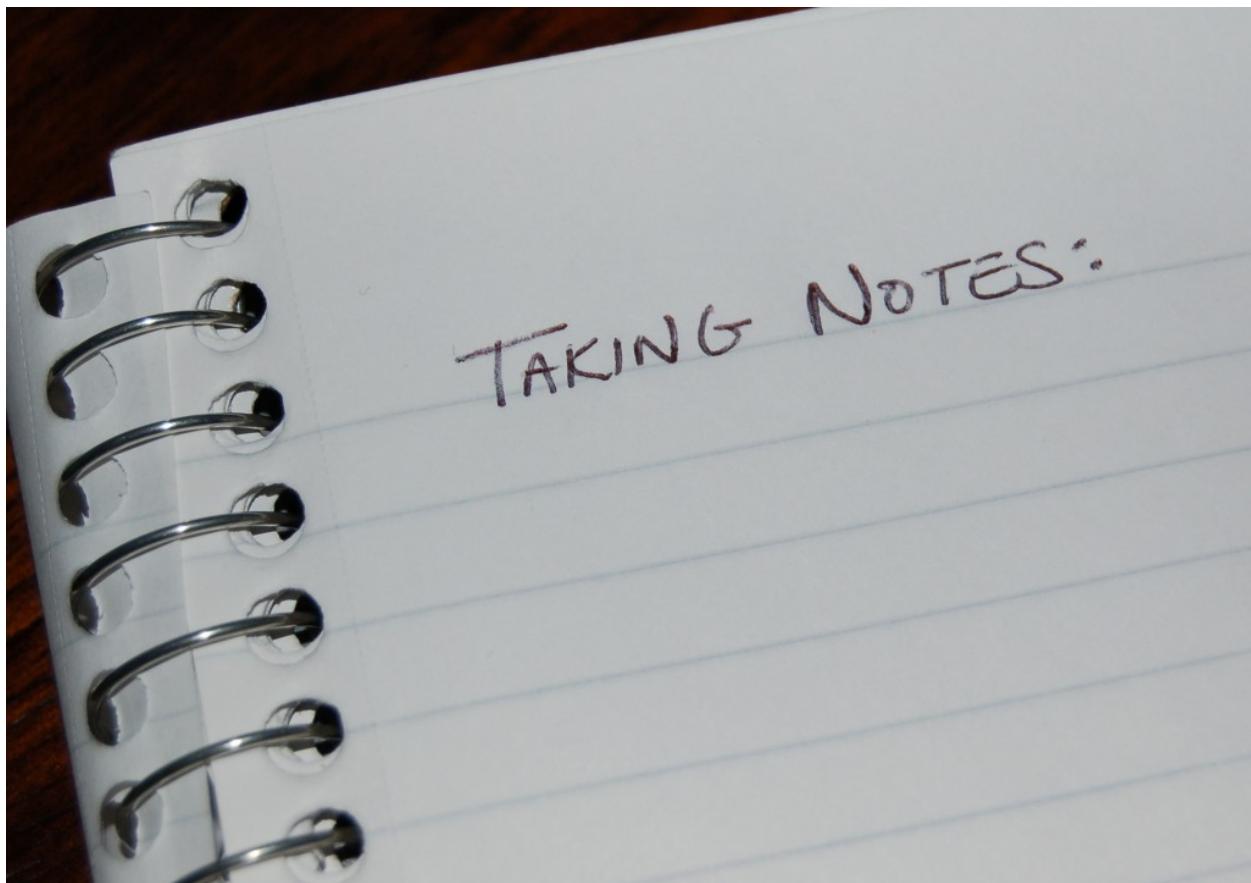
¹⁴⁸<http://thomaswallet.blogspot.com.ar/>

¹⁴⁹<http://www.gdtimes.com/>

¹⁵⁰[http://en.wikipedia.org/wiki/David_Allen_\(author\)](http://en.wikipedia.org/wiki/David_Allen_(author))

¹⁵¹<http://www.yorokobu.es/33-formas-para-seguir-siendo-creativo/>

#2 “Llevá tu cuaderno a todas partes”

**Tomar notas**

Es importante tomar notas, en conferencias, clases, reuniones, workshops o, simplemente, cuando una idea viene a nosotros. Uno nunca sabe cuando la inspiración puede apoderarse de nosotros y hay que estar preparados.

Sin embargo, tomar notas de manera efectiva puede no ser tan simple como parece. Nuestras notas deben ser legibles y, además, captar toda la información relevante para poder luego “seguirle el rastro”.

Algunas sugerencias:

- Anota la fecha de la nota.
- Usa tus propias palabras.
- Revisa las notas tan pronto como puedas luego de la clase/conferencia/reunión, etc.
- Enfatiza aspectos relevantes.
- Anota nombres, fechas, libros y referencias.
- Tacha todo lo que no corresponda.

- Registra cualquier duda que surja sobre el contenido de tu nota para responderla a su debido tiempo.
- Incluye gráficos y/o mapas conceptuales para relacionar conceptos y/o palabras claves.

Mantenerse creativo (3)

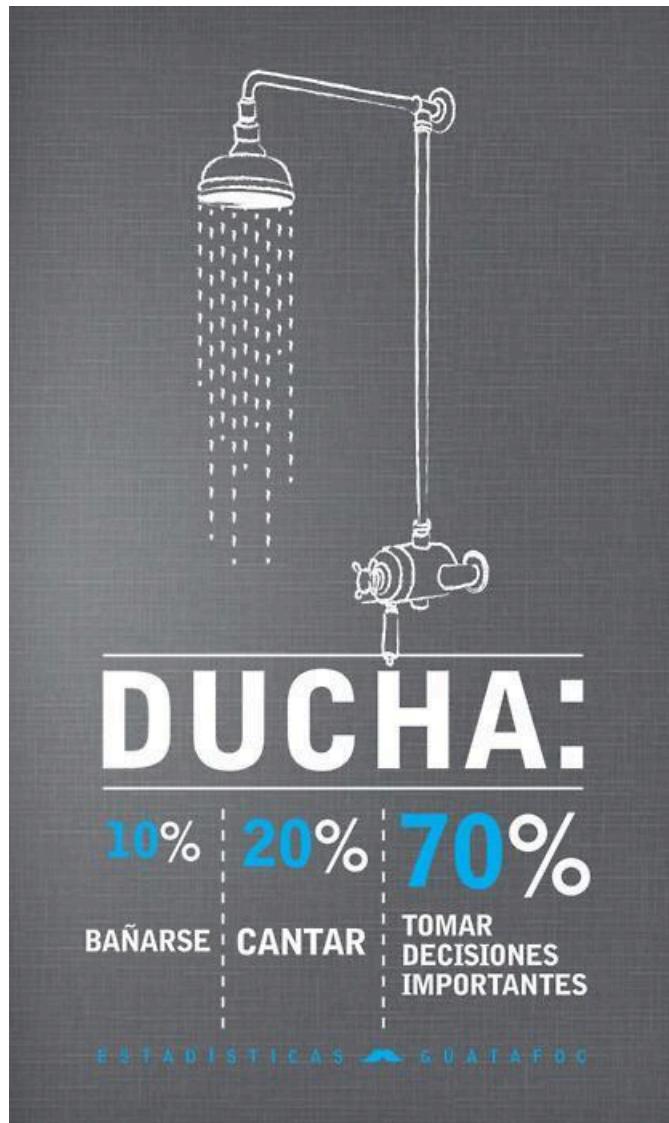
Más de los 33 tips para mantenerse creativo¹⁵².

#3 ¡Cantá en la ducha!

y para que no piensen que estoy loco, hago extensivo este tip con la siguiente pic relacionada. La imagen es de “the cool ruler¹⁵³”

¹⁵²<http://www.yorokobu.es/33-formas-para-seguir-siendo-creativo/>

¹⁵³<http://thecoolruler.blogspot.com.es/>



Cantar en la ducha

Pueben, y después me cuentan...

Mantenerse creativo (4)

#4. Visitá nuevos lugares

Este último fin de semana visité Villa Pehuenia¹⁵⁴. Un centro turístico con cultura milenaria, dotada de bosques de Araucarias Araucanas. Esta aldea de montaña está enclavada en la cordillera de los Andes a la vera de los lagos Aluminé y Moquehue, a 310 Km. de la capital neuquina y a 1200 m.s.n.m. UN LUGAR PERFECTO.

¹⁵⁴<http://www.villapehuenia.gov.ar/inicio.php>

Aquí, mi foto del lugar:



Villa Pehuenia

Fuente: 33 tips para mantenerse creativo¹⁵⁵.

¹⁵⁵<http://www.yorokobu.es/33-formas-para-seguir-siendo-creativo/>